Министерство науки и высшего образования РФ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Уральский государственный педагогический университет»
Институт математики, физики, информатики и технологий
Кафедра информатики, информационных технологий
и методики обучения информатике

МЕТОДИКА ОБУЧЕНИЯ УЧАЩИХСЯ СТАРШИХ КЛАССОВ РЕШЕНИЮ ЗАДАЧ ПО РАСПОЗНАВАНИЮ ОБРАЗОВ В КУРСЕ РОБОТОТЕХНИКИ

Выпускная квалификационная работа по направлению «44.03.01 Педагогическое образование», профиль «Информатика»

Квалификационная работа
Допущена к защите
Зав. кафедрой:
«___» _____2019г.

_____
подпись

Исполнитель:
Димитрова Мария Дмитриевна,
студентка группы ИНФ-1501

_____
подпись

Руководитель:
Шимов Иван Владимирович
старший преподаватель
кафедры ИИТиМОИ

_____
подпись

Екатеринбург – 2019

РЕФЕРАТ

Димитрова М. Д. МЕТОДИКА ОБУЧЕНИЯ УЧАЩИХСЯ СТАРШИХ КЛАССОВ РЕШЕНИЮ ЗАДАЧ ПО РАСПОЗНАВАНИЮ ОБРАЗОВ В КУРСЕ РОБОТОТЕХНИКИ, выпускная квалификационная работа: 72 стр., рис. 21, табл. 3, библ. 31 назв.

Ключевые слова: лабораторный практикум, обучение учащихся старших классов, методика, робототехника, робототехнический комплекс, LEGO Mindstorms EV3, среда программирования, LeJOS, объектно-ориентированный язык программирования Java, веб-камера, распознавание образов, машинное зрение.

Объект исследования: процесс обучения программированию учащихся старших классов.

Цель исследования: разработать лабораторный практикум для обучения старшеклассников решению задач по распознаванию образов в курсе робототехники.

Работа посвящена разработке лабораторного практикума для обучения учащихся старших классов решению задач по распознаванию образов в курсе робототехники. Проводится анализ робототехнических устройств и совместимых с ними веб-камер. Рассматриваются среды программирования для выбранного робототехнического комплекса. Проводится анализ авторских программ в образовательной робототехнике и литературы, посвященной распознаванию образов.

Представлен лабораторный практикум, который состоит из пяти лабораторных работ. Выполнение лабораторных работ позволит учащимся старших классов закреплять веб-камеру на робототехническом устройстве, решать задачи по распознаванию образов и писать программы на языке высокого уровня.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

За последние годы достижения в области робототехники затронули многие сферы человеческой жизни. Традиционно область применения роботов была сосредоточена в промышленном секторе, но за минувшее десятилетие робототехника стала охватывать такие важные сферы жизнедеятельности человека, как строительство, быт, медицина, системы безопасности и т.п. Под термином робототехника понимается прикладная наука, занимающаяся разработкой автоматизированных технических систем [3]. Развитие научных дисциплин и технологий, связанных с робототехникой, позволяет быстро и качественно реализовывать различные робототехнические идеи и проекты в профессиональной деятельности. Развитие автоматизированных технических систем – не исключение. Машинное зрение – один из самых актуальных методов автоматизации процессов с использованием компьютерных технологий и робототехники.

На сегодняшний день наблюдается значительный рост интереса к распознаванию образов. Возможности машинного зрения в области робототехники позволяют роботам решать различные задачи, такие как распознавание объектов и целей, навигация, захват и манипулирование. Чаще для решения задач подобного рода используются методы, основанные на анализе изображений. Работа с камерой может быть рассмотрена как первый шаг в изучении машинного зрения в общеобразовательном учреждении.

Подключение к роботу веб-камеры позволяет расширить круг задач при обучении программированию в старших классах, учащиеся которых, подойдя к высшей ступени обучения в школе, накопили достаточный опыт работы по решению задач с использованием стандартного инструментария в рамках освоения курса робототехники. К тому же, освоив библиотеки машинного зрения и научившись реализовывать программы по распознаванию образов, учащиеся могут стать успешными участниками в соревнованиях по робототехнике, типа AutoNet, направленных на

популяризацию и развитие современных технологий среди молодежи и формированию компетенций, необходимых современному инженеру [17].

Актуальность исследования обусловлена тем, что на сегодняшний день распознавание образов имеет широкое применение и существенно расширяет возможности информационных технологий в сферах безопасности, медицины, промышленности, строительства и т.д. Изучение темы распознавания образов в старших классах значительно повысит карьерные перспективы учеников.

Объект исследования: процесс обучения программированию учащихся старших классов.

Предмет исследования: решение задач по распознаванию образов в курсе робототехники.

Цель исследования: разработать лабораторный практикум для обучения старшеклассников решению задач по распознаванию образов в курсе робототехники.

Задачи исследования:

1. Провести анализ робототехнических устройств и совместимых с ними веб-камер.
2. Провести обзор сред программирования для выбранного робототехнического комплекса.
3. Проанализировать авторские программы в образовательной робототехнике и литературу, посвященную распознаванию образов.
4. Разработать лабораторный практикум для обучения старшеклассников решению задач по распознаванию образов в курсе робототехники и провести апробацию.
5. Разработать методические рекомендации по использованию лабораторного практикума.

ГЛАВА I. РАСПОЗНАВАНИЕ ОБРАЗОВ И КОМПЬЮТЕРНОЕ ЗРЕНИЕ В РОБОТОТЕХНИКЕ

1.1. Аппаратное обеспечение для решения задач по распознаванию образов

Учитывая развитие современных технических средств, ФГОС основного общего образования вносит изменения в содержание современного обучения школьников. Данные изменения связаны с требованиями к материально-техническим условиям реализации основной образовательной программы, одним из которых является обеспечение возможности «проектирования и конструирования, в том числе моделей с цифровым управлением и обратной связью, с использованием конструкторов; управления объектами; программирования». Многие образовательные учреждения реализуют данные требования с помощью внедрения особой образовательной технологии – образовательной робототехники. Одним из ключевых вопросов при реализации данной технологии является подбор конструктора, позволяющего решать современные образовательные задачи [16].

Наиболее популярными робототехническими конструкторами в образовательных учреждениях являются следующие модели:

1. Arduino.
2. Fischertechnik.
3. HUNA-MRT.
4. LEGO Mindstorms.
5. Robotis.
6. TETRIX.
7. TRIK.
8. VEX IQ.

На сегодняшний день наблюдается значительный рост интереса к распознаванию образов в области робототехники. Машинное зрение – один из самых актуальных методов автоматизации процессов с использованием

компьютерных технологий и робототехники. Общую схему работы машинного зрения можно представить, как обработку данных, полученных с камеры, и реакция робототехнической системы на них. Возможности машинного зрения позволяют роботам выполнять различные задачи, такие как распознавание объектов и целей, навигация, захват и манипулирование. Часто для решения подобного рода заданий используются методы, основанные на анализе изображений. Обеспечить робота «зрением» можно, подключив к нему веб-камеру.

Камеру в числе дополнительных комплектующих имеют следующие робототехнические комплексы:

1. Fischertechnik.
2. LEGO Mindstorms.
3. TRIK.

Fischertechnik ROBOTICS

Робототехнический комплекс fischertechnik производятся немецкой фирмой. Для изготовления деталей конструктора используется пластмасса. Платформа fischertechnik включает визуальную среду программирования ROBO Pro. Программирование осуществляется путем составления блок-схем. Современные наборы fischertechnik комплектуются микроконтроллером ROBOTICS TXT (Рис. 1).

Рис. 1. Микроконтроллер ROBOTICS TXT

ROBOTICS TXT обладает следующими характеристиками:

- цветной сенсорный дисплей с диагональю 2.4 дюйма и разрешением 320 x 240 пикселей;
- оперативная память: 128 Мб;
- FLASH-память: 64 Мб;
- поддержка технологии Bluetooth и Wi-Fi;
- инфракрасный приемник для дистанционного управления;
- 8 портов ввода для подключения датчиков;
- 8 портов вывода для исполнительных устройств;
- порт USB–хост;
- слот для карт памяти формата microSD (до 32Гб).

Контроллер TXT работает под управлением операционной системы с ядром Linux. При желании можно загрузить свою ОС с подключаемой карты памяти формата microSD.

Помимо конструкторских деталей и электронных компонентов в наборе присутствует USB-видеокамера (Рис. 2). Используя ее, можно не только передавать видеопоток с робота на компьютер, но и реализовать элементы машинного зрения, связанные с определением цвета, отслеживанием маршрута и движением объектов в той или иной зоне.

Рис. 2. USB-видеокамера, входящая в набор fischertechnik

Основные характеристики камеры:

- матрица: CMOS 1,0 Мпикс;
- разрешение: 1280 x 720 пикселей;

- частота кадров максимальная: 60 Гц при разрешении 320 x 240 пикселей;
- способ фокусировки: ручной;
- встроенный микрофон: имеется;
- интерфейс: USB 2.0;
- длина соединительного кабеля: 80 см.

В инструментальном программном обеспечении ROBO Pro имеется библиотека функций компьютерного зрения и конфигуратор для настройки и наладки. Библиотека компьютерного зрения содержит следующие базовые функций обработки видеокадров:

- вычисление усредненной яркости заданного региона видеокадра;
- поиск последовательности элементов растра одного цвета для заданного региона видеокадра;
- поиск геометрической фигуры, вычисление координат и размеров;
- детектор движения в регионе видеокадра.

Стоимость базового конструктора fischertechnik ROBOTICS TXT Discovery set варьируется в пределах 28500-36000 рублей [21].

LEGO Mindstorms

Впервые робототехнический конструктор LEGO Mindstorms был представлен в 1998 году. В 2006 году вышла вторая версия конструктора – NXT, а в начале 2013 – EV3 (сокращение от Evolution 3).

Таблица 1.

Сравнение версий конструктора LEGO Mindstorms

	EV3	*NXT*
Дисплей	Монохромный LCD, 178x128	Монохромный LCD, 100x64
Процессор	300 МГц Texas Instruments Sitara AM1808 (ARM9)	48 МГц Atmel AT91SAM7S256 (ARM7TDMI)
Память	64 Мб RAM 16 Мб Flash Слот microSDHC (до 32 Гб)	64 Кб RAM 256 Кб Flash
USB-хост	+	–
Wi-Fi	+	–
Bluetooth	+	+

Из приведенной таблицы можно сделать вывод, что третья версия микроконтроллера LEGO Mindstorms обладает наиболее лучшими характеристиками. В дальнейшем исследовании будем рассматривать робототехнический конструктор LEGO Mindstorms EV3.

Набор LEGO Education Mindstorms EV3 поставляется в пластиковой коробке с крышкой, имеет два лотка для сортировки деталей. С помощью этого конструктора можно собрать как колесные, так и гусеничные платформы. В комплект входит печатное руководство по хранению деталей и инструкция по сборке базовой платформы. Робототехнический комплекс оснащен оптимальным набором датчиков: два датчика касания, ультразвуковой датчик, датчик цвета/освещенности, гироскоп и тремя сервомоторами.

Рис. 3. Микроконтроллер EV3

Интеллектуальный блок EV3 (Рис. 3) служит центром управления и энергетической станцией для робота и имеет следующие функциональные элементы:

- многофункциональный монохромный дисплей;
- оперативная память: 64 Мб;
- FLASH-память: 16 Мб;
- шестикнопочный интерфейс управления с функцией изменения подсветки для индикации режима работы;
- 4 порта ввода для подключения датчиков;

- 4 порта вывода для выполнения команд;
- поддержка Wi-Fi и Bluetooth;
- разъём miniUSB для подключения EV3 к компьютеру;
- порт USB–хост;
- слот для карт памяти формата microSD (до 32Гб);
- встроенный динамик.

Также программируемый блок оснащен автономным питанием и работает под операционной системой Linux.

LEGO Mindstorms – самая популярная модель конструируемых роботов в общеобразовательных учреждениях, именно этими конструкторами оснащено большинство кружков робототехники. Это отличный вариант и по соотношению цена/качество, и по соотношению доступность/возможности.

Стоимость набора LEGO Education Mindstorms EV3 варьируется в пределах 25000-30000 рублей [23].

Для конструкторов Mindstorms EV3 дополнительным комплектующим является камера PIXY (Рис. 4).

Рис. 4. Видеокамера PIXY

Видеокамера PIXY производится компанией Charmed Labs и предоставляет простую возможность реализации элементов машинного зрения. В отличие от большинства камер, PIXY выполняет обработку изображения прямо на своем борту, освобождая мощности микроконтроллера для решения других задач. Камера подключается

напрямую к контроллеру с помощью прилагаемого кабеля и интегрируется в программную среду LEGO.

Работа с камерой в среде LEGO Mindstorms EV3 напоминает работу со стандартными датчиками и не требует дополнительного программирования. PIXY представляет собой программируемый встроенный датчик компьютерного зрения.

Для обнаружения объектов камера использует алгоритм фильтрации, основанный на цветах. PIXY вычисляет оттенок и насыщенность каждого пикселя изображения и использует в качестве параметров первичной фильтрации. Видеокамера помнит до 7 подписей разных цветов (красный, оранжевый, желтый, зеленый, голубой, синий, фиолетовый), при необходимости использования большего количества цветов необходимо использовать коды цвета.

Основные характеристики:

- датчик изображения: OnSemi MT9M114;
- разрешение: 1296×976 пикселей с интегрированным процессором потока изображений;
- поле обзора объектива: 60 градусов по горизонтали, 40 градусов по вертикали;
- процессор: двухъядерный NXP LPC4330 Arm Cortex;
- порт: microUSB;
- энергопотребление: 140 мА;
- размеры: 42 х 38 х 15 мм [26].

Однако камера не представлена в стандартном наборе и приобретается отдельно, также подключение данного устройства занимает входной порт, что уменьшает общее число подключаемых датчиков.

TRIK

Детали российского конструктора TRIK изготовлены из металла. Набор укомплектован мощными серводвигателями в виду веса металлической конструкции. Разница между наборами, отличающихся в цене, заключается в

количестве датчиков и деталей. В каждом наборе есть контроллер TRIK, видеокамера, микрофон и пластиковая коробка для хранения деталей.

Главной особенностью набора является контроллер TRIK (Рис. 5) под управлением операционной системы Linux.

Рис. 5. Контроллер TRIK

Технические характеристики контроллера:

- центральный процессор ARM, 375 МГц;
- цветной сенсорный LCD монитор с диагональю 2.4 дюйма и разрешением 320 x 240 пикселей;
- оперативная память: 256 Мбайт;
- FLASH-память: 16 Мбайт;
- поддержка технологии Bluetooth и Wi-Fi;
- порт USB–хост;
- слот для карт памяти формата microSD;
- 4 порта двигателей с индивидуальной аппаратной защитой от перегрузки по току;
- 19 сигнальных портов общего назначения;
- 2 входа для интерфейсов видео сенсоров;
- встроенный динамик;
- 2-цветный светодиодный индикатор.

Набор конструктора TRIK комплектуется видеомодулем OV7670 (Рис. 6).

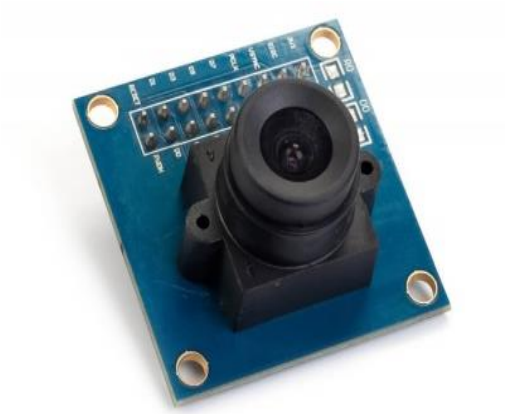


Рис. 6. Видеомодуль конструктора TRIK

Видеокамера обладает следующими характеристиками:

1. Количество пикселей: 0,3 Мп.
2. Интерфейс: Standard SCCB interface compatible with I2C interface.
3. Поддерживает: VGA, CIF.
4. Разрешение матрицы: 640 x 480 пикселей.
5. Угол обзора: 25 градусов.
6. Частота дискретизации: 50/60 Гц.
7. Скорость записи: максимум 30 кадр/с.

Стоимость минимального образовательного набора TRIK – 57800 рублей, стандартного образовательного набора – 75000 рублей [30].

Несмотря на то, что контроллеры fischertechnik и TRIK обладают лучшими характеристиками и камера уже входит в набор, в дальнейшем исследование будет основано на конструкторе LEGO Mindstorms, так как данный конструктор является самым распространенным в общеобразовательных учреждениях. К тому же данный робототехнический комплекс обладает более низкой ценой.

Возникает следующая проблема: камера PIXY не входит в стандартный набор, а ее приобретение обойдется в районе 60$, закупка оригинальных камер для конструкторов LEGO может сильно ударить по бюджету образовательного учреждения.

В поисках аналога, была найдена веб-камера Vision Subsystem, но ее стоимость в среднем составляет около 150$.

Аналог PIXY, превышающий ее стоимость почти в три раза, имеет следующие характеристики:

- подключение к EV3 через порт датчика;
- подключение к компьютеру с помощью интерфейса USB;
- скорость записи 30 кадр/с;
- одновременное распознавание до 8 объектов;
- регулируемый фокус объектива;
- объектив с встроенным инфракрасным блокирующим фильтром;
- предоставление статистики отслеживания в реальном времени;
- автономная работа на микроконтроллере EV3 [31].

Реальным решением данной проблемы является приобретение камеры, обладающей более низкой стоимостью. Для работы с EV3 подходят многие простые и недорогие устройства захвата изображения, реже подходят те камеры, которые имеют встроенный web-сервер.

В связи с отсутствием рекомендованного списка совместимых веб-камер с модулем EV3, перечень возможных сторонних камер был составлен в соответствии с характеристиками устройств захвата изображения, представленных в других робототехнических наборах (Таблица 2).

Таблица 2.

Характеристики веб-камер

	Logitech Carl Zeiss Tessar	*Genius WideCam F100*	*A4Tech PK-750G*	*Logitech WebCam C170*
Разрешение матрицы	2 Мп	2 МП	0,3 Мп	0,3 Мп
Разрешение матрицы (без интерполяции)	1920 x 1080	1280 x 720	640 x 480	640 x 480
Фоторазрешение в режиме интерполяции	15 Мп	12 Мп	16 Мп	5 Мп
Интерфейс	USB2.0	USB 2.0	USB2.0	USB2.0
Запись видео высокой четкости	1080p	1080p	720p	720p
Скорость записи видео	30 кад/сек	30 кад/сек	30 кад/сек	30 кад/сек
Цена	5 500 рублей	2 855 рублей	1 150 рублей	1 030 рублей

Подключение сторонней камеры к модулю EV3 сопровождается следующей проблемой: среда разработки LEGO Mindstorms Education EV3 не

поддерживает работу с внешними камерами. Решением данного вопроса является программирование робота на языках, позволяющих реализовать такую возможность. Проведем анализ различных сред программирования в следующем параграфе.

1.2. Программное обеспечение для решения задач по распознаванию образов

Стандартное программное обеспечение для блока EV3 знакомит учеников с основами программирования, однако со временем возникает потребность в использовании более профессиональных языков для выполнения сложных задач. Для блока EV3 разработано более десятка различных сред программирования, которые позволяют расширить его возможности, скорость работы и т.д.

Средой программирования называется комплекс программ для разработки и отладки программ. Классифицировать все среды программирования можно на визуальные и невизуальные. При помощи графического программирования программа создается путем манипулирования графическими объектами. Текстовый язык программирования – формальный язык, предназначенный для записи компьютерных программ [11].

Обратим внимание на то, что обучение решению задач по распознаванию образов организовано для учеников старших классов. Обучение старшеклассников текстовому языку программирования в рамках курса робототехники отвечает требованиям ФГОС к предметным результатам освоения информатики. К тому же в современном обществе обладание навыком программирования на объектно-ориентированных языках высоко ценится среди выпускников общеобразовательных учреждений. Поэтому при выборе программных средств отдадим предпочтение невизуальным средам программирования.

Выделим критерии для сравнения средств программирования:

- язык;

- режим отладки;
- симулятор;
- лицензия;
- методическая поддержка;
- возможности программных структур;
- поддержка библиотек машинного зрения (основной критерий сравнения).

Программировать микроконтроллер EV3 можно с помощью различных альтернативных сред разработки. Платные среды программирования представляют LabVIEW и RobotC. Из бесплатных можно обратить внимание на ev3dev, поддерживающий языки C++, Lua, Node.js, Python, Google Go, C и Clojure. Установка виртуальной Java-машины leJOS откроет возможность программировать на Java или Scratch. Для поклонников .NET существуют проекты LEGO Mindstorms EV3 API и MonoBrick. Для пользователей Microsoft Small Basic есть расширение EV3 Basic. Рассмотрим самые популярные из них.

LabVIEW

LabVIEW – проприетарная графическая среда разработки программного обеспечения на визуальном языке потоков данных G. Язык G моделирует процесс вычислений, в котором между блоками задаются связи, отличающиеся типом передаваемых через них данных. Выходные данные одного блока могут служить входными данными для другого. Блоки начинают исполняться, когда имеют данные на всех входах. В случае, когда таких блоков несколько, они исполняются параллельно.

Среда поддерживает большое количество аппаратных платформ и предоставляет десятки библиотек, которые содержат средства для работы со сложными математическими конструкциями, алгоритмы компьютерного зрения, средства взаимодействия с системами управления базами данных и т.д. LabVIEW позволяет интерпретировать программы и генерировать по ним код для автономного запуска программ на устройстве.

Алгоритмические конструкции поддержаны в полном объеме, однако модель исполнения является потоковой, что может быть весьма удобным для программирования многих алгоритмических конструкций, но сложным для изучения и понимания. Создаваемые в среде программы довольно часто получаются громоздкими и трудно читаемыми, особенно если создаются неопытными пользователями.

В среде имеются компоненты, позволяющие применить LabVIEW в образовательных целях. В частности, среда поддерживает робототехнические конструкторы LEGO Mindstorms. Система LabVIEW достаточно сложна для изучения, отмечается, что на непосредственное освоение среды тратится большое количество времени, так как она содержит более сотни блоков. К тому же среда обладает дорогостоящей лицензией [29].

Подобные факторы делают довольно редкими применения LabVIEW в школьном образовании, гораздо чаще для этого используются адаптации, основанные на базе данной среды. В нашей работе мы рассмотрим адаптацию для конструктора LEGO Mindstorms EV3 – EV3-G.

EV3-G

Среда EV3-G – свободное программное обеспечение, поставляемое в комплекте с конструктором LEGO Mindstorms EV3. EV3-G основана на базе LabVIEW и позволяет программировать микроконтроллер на языке G.

Блоки, из которых создается программа, автоматически соединяются друг с другом, имеют понятное и удобное для редактирования графическое представление. Все блоки в среде делятся на 6 групп, принадлежность к группе определяется цветом. Палитра блоков делится на вкладки: действие, управление операторами, датчик, операции с данными, дополнения и собственные блоки пользователя.

EV3-G позволяет формировать достаточно сложные алгоритмы и реализовывать многие структуры программирования. Среда поддерживает возможность просмотра, сбора, обработки, анализа и прогнозирования

измерений, получаемых с датчиков. EV3-G автоматически определяет к какому порту подключен датчик, благодаря функции auto-id.

Особое внимание заслуживает возможность подключения модуля машинного зрения. Совместно с камерой PIXY среда LEGO Mindstorms Education EV3 позволяет реализовывать сложнейшие технические задачи выделения геометрических форм, определения объектов, заданных эталонами, проведения бесконтактных измерений, считывания и распознавания символьных изображений и т.д.

Среда включает в себя большую коллекцию заданий по программированию, мультимедийные справочные материалы, инструкции и интерактивные примеры.

Основные достоинства EV3-G:

- система является бесплатной;
- интуитивно понятный и удобный интерфейс;
- возможность интеграции модуля машинного зрения;
- мощная методическая поддержка;
- возможность реализации сложных алгоритмов.

К недостаткам среды можно отнести:

- отсутствие возможности отладить программу на симуляторе;
- громоздкость диаграмм при создании больших и сложных программ;
- при написании сложных и разветвленных алгоритмов программное обеспечение сильно грузит систему [5, 19].

EV3 Basic

EV3 Basic – свободно распространяемый текстовый язык программирования для платформы LEGO Mindstorms EV3, являющийся расширением для среды Microsoft Small Basic. По уровню сложности данный язык не очень отличается от рассмотренного ранее EV3-G, но в тоже время обладает рядом преимуществ, присущих текстовым языкам программирования.

Microsoft Small Basic EV3 разработан специально для обучения программированию, следовательно, очень прост в изучении. EV3 Basic наследует все библиотеки Small Basic, которые позволяют реализовывать стандартные программы из основного курса робототехники такие, как взаимодействие с моторами, датчиками, динамиком, экраном и кнопками модуля EV3. Готовые программы могут быть запущены в режиме онлайн, для загрузки программы в память микроконтроллера, необходимо использовать отдельную утилиту EV3 Explorer.

Программы языка Microsoft Small Basic состоят из связки командных предложений, которые называются функциями. Каждая функция состоит из объекта, метода и параметров. Строки программы представляют собой предложения, которые являются командами для исполнителя.

Данная среда позволяет постепенно изучать основные возможности объектно-ориентированных языков программирования, а также знакомит с синтаксисом текстовых языков. Также среда оснащена окном справки, в котором отражаются подсказки по синтаксису языка.

Основными достоинствами EV3 Basic являются:

- свободное программное обеспечение;
- русифицированная среда;
- методическая поддержка и окно справки;
- интуитивно понятный интерфейс;
- простота освоения.

К недостаткам можно отнести то, что среда не поддерживает библиотеки компьютерного зрения [13].

Таким образом, Microsoft Small Basic EV3 отлично подходит для обучения основам робототехники школьников младших и средних классов, которым только предстоит знакомство с объектно-ориентированным программированием. В нашем случае уровень сложности и возможности среды EV3 Basic не удовлетворяют требованиям обучения старшеклассников

решению задач по распознаванию образов, поэтому в дальнейшем исследовании мы не будем рассматривать данную среду.

Scratch

Scratch является визуальной кроссплатформенной средой программирования с открытым исходным кодом, разработанной исследовательской группой Массачусетского технологического института. Возможность программировать робототехнический комплекс LEGO Mindstorms EV3 осуществляется посредством установки расширений на базе Scratch и дополнительного программного обеспечения.

Встроенная система помощи и подсказок в Scratch в настоящее время не переведена на русский язык. Тем не менее система содержит изображения и анимацию, которые доступно иллюстрируют принципы работы в среде и назначения блоков.

Программирование в среде Scratch осуществляется посредством соединения объектов, которые называются спрайты. Действие программы происходит в отдельном окне с центром координат в середине сцены. Для программирования сценариев используется метод drag-and-drop. Функциональные возможности блоков различаются цветом и делятся на 10 категорий: движение, события, внешность, управление, звук, сенсоры, перо, операторы, данные и другие блоки.

К плюсам программирования в среде Scratch можно отнести:

- интуитивный пользовательский интерфейс;
- возможность отладки удаленного управления роботом с компьютера;
- свободное программное обеспечение;
- открытый исходный код;
- методическая поддержка среды;
- возможность исполнения программы на виртуальном спрайте.

К отрицательным сторонам можно отнести:

- отсутствие некоторых алгоритмических аспектов;

- отсутствие возможности генерации читаемого кода по визуальной модели;
- требуется перепрошивка микроконтроллера;
- отсутствие библиотек машинного зрения [27].

В связи с последним недостатком в дальнейшем исследовании среда Scratch не рассматривается.

Open Roberta Lab

Open Roberta – облачная среда визуального программирования роботов LEGO Mindstorms EV3, основанная на базе Scratch. Программирование в Open Roberta осуществляется на визуальном языке NEPO. Главная цель проекта – преодоление технического и профессионального барьеров как для преподавателей, так и для учеников, поэтому среда является бесплатной и имеет открытый исходный код.

Так как платформа является облачной, её можно использовать в любое время с любого устройства, где есть браузер и подключение к Интернету. Для хранения результатов работы необходимо зарегистрироваться в лаборатории Open Roberta.

Важной особенностью Open Roberta является наличие двумерного симулятора робота, интегрированного со средой. Среда позволяет настраивать конфигурации робота для корректной работы программы. В Open Roberta поддерживаются только роботы с двумя ведущими колесами, для которых задается диаметр и расстояние между ними. Также в настройках конфигурации указываются подключенные к портам датчики и моторы.

После отладки программы на симуляторе можно передать ее на реального робота. В этот момент пользователи могут столкнуться с самым большим недостатком проекта, так как коммуникации с реальным роботом затруднены. Во-первых, для начала работы необходимо перепрошить робота. Во-вторых, единственный способ общения среды Open Roberta с микроконтроллером состоит во встраивании робота в общую с компьютером

TCP-сеть посредством Wi-Fi модуля, при этом среда высылает Ajax-запросы на удаленный сервер, а робот в этом случае действует как HTTP-клиент.

Достоинства:

- методическая поддержка среды, однако только англоязычная;
- свободное программное обеспечение;
- открытый исходный код;
- не требует установки на компьютер;
- проверка на симуляторе.

Недостатки:

- отсутствие средства генерации кода;
- интерфейс не локализован на русский язык;
- не реализованы возможности машинного зрения [22].

Несмотря на значительные достоинства среды Open Roberta Lab, использовать ее в целях обучения компьютерному зрению невозможно, поэтому в дальнейшем исследовании данная среда рассматриваться не будет.

TRIK Studio

TRIK Studio – среда визуального и невизуального программирования. TRIK Studio поддерживает множество языков программирования, но, как правило, наиболее часто используется – визуальный.

Программа в TRIK Studio составляется из блоков и стрелок, вместе описывающих поток управления алгоритма. Математические выражения, условия на развилках, значения свойств описываются на встроенном текстовом языке – Lua.

Среда предоставляет три режима работы с конструктором: режим интерпретации, режим автономного исполнения и режим отладки на симуляторе.

В режиме интерпретации программа исполняется на компьютере с отправкой команд роботу по низкоуровневому протоколу. Значения всех переменных во время интерпретации могут быть просмотрены в

соответствующем окне, а также можно отслеживать графики показаний датчиков, строящиеся в реальном времени.

В режиме автономного исполнения среда генерирует код, после компилирует и загружает его по низкоуровневому протоколу на робота и запускает его на исполнение. Код генерируется в читаемом виде, он может быть открыт и отредактирован во встроенном текстовом редакторе с подсветкой синтаксиса и автоматическим дополнением.

В режиме симуляции программа будет выполнена на двумерной модели робота, открываемой внутри окна среды. Двумерный симулятор позволяет пользователю нарисовать произвольную модель мира, состоящую из стенок, регионов и цветных элементов. К примеру, могут быть нарисованы все стандартные поля и полосы препятствий, используемые в спортивной робототехнике. Далее указывается, какие датчики подключены к роботу, их пространственное положение и ориентация. Затем программа может быть исполнена на нарисованной модели мира, при этом можно отслеживать значения переменных и графики значений сенсоров.

Несмотря на то, что основная часть работы по написанию программы выполняется в визуальном редакторе, детям все же необходимо взаимодействовать с элементами текстового языка. Это может быть рассмотрено и как преимущество (благоприятные условия для плавного перехода на текстовые языки) и как недостаток (значительное повышение порога вхождения в среду).

К достоинствам среды TRIK Studio можно отнести:

- свободное программное обеспечение;
- открытый исходный код;
- наличие справочной системы;
- русифицированная среда;
- возможность отладки программы в режиме реального времени.

Недостатки:

- отсутствие методической поддержки;

- возможности компьютерного зрения не реализованы;
- отсутствие многих алгоритмических аспектов [6, 7].

Как и многие рассмотренные ранее среды программирования, TRIK Studio не реализует возможности компьютерного зрения, поэтому не участвует в дальнейшем исследовании.

RobotC

RobotC – кроссплатформенная среда программирования, разработанная специально для образовательной робототехники. RobotC принадлежит к семейству C-подобных языков.

В среде представлены два режима программирования: базовый и расширенный. В соответствии с этим среда программирования поддерживает два языка: RobotC и Natural Language, являющийся плавным переходом от визуального программирования к текстовым блокам.

RobotC схожа со средой программирования Visual Studio и имеет англоязычный интерфейс. В среде присутствует мощный интерактивный отладчик, способный функционировать в режиме реального времени.

При помощи среды программирования RobotC можно создавать эффективные программы с использованием сложных математических выражений.

RobotC является проприетарным программным обеспечением с пробной полнофункциональной 30-дневной версией.

Достоинства:

- методическая поддержка среды;
- интерактивный отладчик в режиме реального времени;
- поддержка сложных математических выражений и алгоритмических аспектов.

Недостатки:

- проприетарное программное обеспечение;
- нерусифицированная среда;
- отсутствие библиотек машинного зрения;

- требуется перепрошивка микроконтроллера [28].

RobotC не удовлетворяет основному критерию – в среде не реализована поддержка библиотек машинного зрения, следовательно, в дальнейшем исследовании среда рассматриваться не будет.

Python

На сегодняшний день язык программирования Python пользуется большой популярностью среди профессиональных программистов. Python имеет небольшое синтаксическое ядро и объемную стандартную библиотеку функций.

Для того, чтобы программировать EV3 на Python необходимо установить прошивку ev3dev на модуль микроконтроллера. Ev3dev является системой Debian Linux и позволяет программировать на языках C++, Lua, Node.js и Python.

Писать программы на языке Python можно в любом текстовом редакторе. Удобнее всего использовать редакторы с подсветкой синтаксиса, такие как Notepad++, Atom или Vim. Достаточно лишь создать файл с расширением *py*. Созданный в программе файл можно передать в память микроконтроллера при помощи программы PuTTY. Эта программа помогает установить SSH соединение операционной системы Windows с роботом.

Также для Python существует несколько сред разработок с возможностью отладки программы, но все они платные, кроме комбинации из среды разработки Eclipse и плагина PyDev.

Важно обратить внимание на то, что Python поддерживает библиотеку алгоритмов компьютерного зрения OpenCV. Библиотека имеет открытый исходный код и доступна для многих современных языков программирования. OpenCV имеет богатый функционал, такой как основные структуры данных, алгоритмы обработки изображений, базовые алгоритмы компьютерного зрения, ввод и вывод изображений и видео, алгоритмы детекции человеческих лиц, поиска стереосоответствия, оптического потока и т.д.

Достоинства:

- программирование на современном объектно-ориентированном языке;
- возможность полноценной отладки программы;
- свободное программное обеспечение;
- широкие возможности использования математических аспектов и структур программирования;
- поддержка библиотеки компьютерного зрения.

К недостаткам можно отнести лишь необходимость перепрошивки микроконтроллера [20].

LeJOS

LeJOS – это Java-совместимая программная среда, разработанная для робототехнических конструкторов LEGO Mindstorms. Среда предоставляет возможность использовать объектно-ориентированный язык Java, что в свою очередь расширяет стандартные возможности EV3, предоставляя полный набор функций JVM через операционную систему Linux. Например, использование двумерных массивов, синхронизацию, исключения, рекурсии и т.д.

Программировать на языке Java можно в среде разработки Eclipse, предварительно подключив к ней библиотеку leJOS. Передавать готовую программу можно через USB-провод, Bluetooth или Wi-Fi.

Важной особенностью является то, что в среду leJOS можно интегрировать библиотеку компьютерного зрения OpenCV, возможности которой были описаны ранее.

Преимущества программирования в LEGO Mindstorms с использованием прошивки leJOS:

- программирование на объектно-ориентированном языке Java со всеми его возможностями;
- возможность создания приложений для смартфонов, планшетов и компьютеров для удаленного взаимодействия с роботом;

- полноценная отладка кода при подключении специального плагина;
- возможность подключения сторонних датчиков и моторов;
- свободное программное обеспечение;
- поддержка библиотеки компьютерного зрения.

Недостатком программирования робота LEGO Mindstorms EV3 на языке Java является необходимость перепрошивки микроконтроллера и полное отсутствие методической поддержки [18, 24].

Обобщим информацию, изложенную выше, и сравним среды программирования, позволяющие реализовать компьютерное зрение, в Таблица 3.

Таблица 3.

Сравнение сред программирования

	EV3-G	*Ev3dev*	*LeJOS*
Язык	Графический	Python	Java
Режим отладки	–	+	+
Симулятор	–	–	–
Лицензия	В свободном доступе	В свободном доступе	В свободном доступе
Методическая поддержка	+	–	–
Возможности программных структур	Ограниченные	В полном объеме	В полном объеме
Компьютерное зрение	+	+	+

Стандартная среда разработки LEGO Mindstorms EV3 не только уступает альтернативным средам по возможностям, но и ограничивает выбор подключаемой к микроконтроллеру веб-камеры. В отличие от предлагаемого компанией Charmed Labs способа наделения робота компьютерным зрением с помощью камеры Pixy, у которой для обработки изображений используется свой процессор, альтернативные среды дарят возможность подключить обычную веб-камеру к USB-порту программируемого блока EV3. К тому же изучение объектно-ориентированного языка программирования может значительно повысить карьерные перспективы учеников, в то время как

обучение программированию EV3 с помощью стандартной системы на основе блоков будет иметь гораздо меньшую профессиональную ценность.

Сами прошивки LeJOS и ev3dev наделены равными возможностями, главное отличие заключается в используемом языке программирования. Мы остановились на языке Java, так как этот язык программирования лежит в основе лабораторного практикума по изучению невизуальной среды программирования LeJOS, продолжение которого является результатом нашего исследования.

1.3. Обзор методических комплексов по решению задач с распознаванием образов

На сегодняшний день компания LEGO является ведущим мировым производителем детских конструкторов. В 1980 году компанией было создано подразделение Education с целью разработки новых образовательных технологий и производства сопровождающей продукции для школ, дошкольных учреждений и учреждений дополнительного образования. LEGO Education удалось разработать целостную концепцию обучения, помимо самих конструкторов, компания предлагает пособия для учителей, рабочие тетради, справочники и программное обеспечение.

Большой вклад в создание роботов серии LEGO Mindstorms внёс профессор Массачусетского технологического института Сеймур Пейперт, основатель MIT Media Lab. В 1988 году Пейперт был назван LEGO-профессором исследований по обучению, а название своего микроконтроллера LEGO позаимствовало у его книги «MINDSTORMS: children, computers, and powerful ideas» [25].

На основе трудов профессора Пейперта, оказавших большое влияние на современное представление о знании и приобретении опыта, построены многие образовательные программы. Его исследования показали, что в программах с участием роботов учащиеся осваивают метакогнитивные навыки, в особенности, в области креативного и критического мышления. Данная форма обучения обозначается специалистами как

«конструкционизм». Согласно данной концепции, эффективному обучению учащихся способствует активное самостоятельное конструирование знаний в процессе реализации собственных идей. Используя данный педагогический метод в процессе освоения курса робототехники, можно получить действенный способ обучения на собственном практическом опыте учащихся [10].

Признание активной роли ученика привело к изменению представлений о взаимодействии участников образовательного процесса. Традиционный информационно-объяснительный подход к построению образования, когда большой объем знаний дается учителем в готовом виде, без опоры на поисковую и самостоятельную работу обучающихся, сменился на компетентностный. Учение стало рассматриваться, как сотрудничество – совместная работа учителя и учеников в ходе овладения знаниями и решения проблем. Данный подход реализован в федеральном государственном образовательном стандарте (ФГОС).

ФГОС и Примерная программа по Информатике и ИКТ (раздел «Алгоритмизация и программирование») ставит следующие цели и задачи:

- знать основные свойства алгоритмов, типы алгоритмических конструкций: следование, ветвление, цикл, понятие вспомогательного алгоритма;
- уметь использовать алгоритмические конструкции, выполнять и строить простые алгоритмы, выполнять базовые операции над объектами: числами, списками, деревьями;
- использовать приобретенные знания и умения в практической деятельности [14].

Несмотря на четко сформулированные цели и поставленные задачи перед нами стоит серьёзная проблема – отсутствие проработанных учебных программ и учебных материалов для педагогов. В связи с тем, что в основном робототехника распространена в области дополнительного образования, зачастую строго прописанных учебных программ не требуется.

Отсюда следует вывод, что основные усилия должны быть приложены к разработке не столько нового аппаратного или программного обеспечения для занятий робототехникой, сколько к разработке учебных материалов и программ для её изучения.

На сегодняшний день ведется активная работа по подготовке учителей в области использования инновационных технологий. К наборам серии LEGO Mindstorms выпущено значительное количество сопутствующих учебных материалов. В отечественной педагогике разрабатываются учебные курсы по робототехнике, как с использованием локализованных материалов LEGO Education, так и на базе собственных исследований. Среди таких можно выделить труды следующих авторов: Белиовская Л.Г., Злаказов А.С., Горшков Г.А., Шевалдина С.Г., Федосов Л.Ю., Филиппов С.А., Чехлова А.В., Якушин С.А., Копосов Д.Г., Овсяницкая Л.Ю., Васильев А. Д.

Существующие учебные курсы и пособия по робототехнике можно разделить на три группы. Первая группа – курсы, основанные на методиках проектной деятельности. Данный подход поддерживается компанией LEGO. В России локализацией таких материалов занимается Институт Новых Технологий. Вторая группа ориентирована на выполнение задач олимпиадного уровня и подготовку к соревнованиям по робототехнике, которые на сегодняшний день пользуются большой популярностью. В данных курсах особый акцент делается на обучение учителей тому, как готовить детей к подобным соревнованиям и решать конкретные олимпиадные задания. Третья группа – курсы по программированию. На сегодняшний день авторы большинства пособий подробно разбирают основы программирования в графических средах, но методических разработок по основам программирования на языках высокого уровня крайне мало [9].

Проведем анализ трех разных авторских учебных программ дополнительного образования: Филиппов С. А. «Робототехника: конструирование и программирование», Копосов Д. Г. «Первый шаг в робототехнике» и Васильев А. Д. «Робототехника».

Рассмотрим образовательную **программу Филиппова С.А.** «Робототехника: конструирование и программирование». Программа рассчитана на детей двух возрастных групп 10-13 и 14-17 лет. Некоторые темы взаимосвязаны со школьным курсом и могут с одной стороны служить пропедевтикой, с другой стороны опираться на него.

Краткое содержание программы. Программа рассчитана на трёхгодичный цикл обучения. В первый год учащиеся проходят курс конструирования, построения механизмов с электроприводом, а также знакомятся с основами программирования контроллеров базового набора. Во второй год учащиеся изучают пневматику, возобновляемые источники энергии, сложные механизмы и всевозможные датчики для микроконтроллеров. Программирование в графической инженерной среде изучается углубленно. Происходит знакомство с программированием виртуальных роботов на языке программирования, схожем с Си. На третий год учащиеся изучают основы теории автоматического управления, интеллектуальные и командные игры роботов, строят роботов-андройдов, а также занимаются творческими и исследовательскими проектами.

Режим занятий: занятия проводятся 2 раза в неделю по 2 учебных часа (144 часа) в первый и второй год обучения и 3 раза в неделю в третий год обучения (216 часов) [15].

Программа Копосова Д.Г. «Первый шаг в робототехнике» носит практико-ориентированный характер: большая часть учебного времени затрачивается на сборки моделей роботов и их программирование. Занятия основаны на практическом выходе, при котором ученик активно вовлечен в свой собственный учебный процесс. Программа рассчитана на учащихся в возрасте от 10 до 16 лет и предполагает четырёхгодичный цикл обучения.

Краткое содержание программы. В первый год учащиеся знакомятся с простейшими основами механики и робототехники, основными видами конструкций и соединений деталей. К концу второго года обучающиеся будут свободно работать в среде LEGO Mindstorms, знать основные приемы

работы с алгоритмическими конструкциями, смогут создавать стандартные модели роботов по образцу. На третий год обучения ученики будут иметь целостное представление о мире техники, разрабатывать творческие модели роботов. К концу четвертого года обучения учащиеся освоят основы механики, робототехники и программирования в среде LEGO Mindstorms и смогут создавать сложные модели роботов.

Режим занятий: для учащихся 1 года обучения 2 раза в неделю по 2 часа (144 часа); для учащихся 2-4 года обучения 2 раза в неделю по 3 часа (216 часов) [4].

Программа Васильева А.Д. «Робототехника» позволяет построить интегрированный курс, сопряженный со смежными направлениями, напрямую выводящий на свободное манипулирование конструкционными и электронными компонентами. Программа в больше мере направлена на организацию коллективной работы учащихся в возрасте от 11 до 17 лет. Срок реализации программы – 2 года.

Краткое содержание программы. Первый год обучения посвящен вхождению в сферу робототехники и профориентации. Воспитанник получает первый опыт командной работы и коллективной ответственности за результат. Второй год обучения призван обучить навыкам управления робототехническими устройствами. В наибольшей степени здесь формируется умение строить управление автономных модулей на основе различной реализации программного управления. Значительную роль начинают играть соревнования на преодоление сложной геометрии трассы и соревнования по международным правилам, что позволяет удержать заинтересованность ребенка в процессе изучения сложного материала. Командная работа на данном этапе должна стать для воспитанника естественной формой деятельности.

Режим занятий: занятия проводятся 1 раз в неделю по 2 учебных часа (68 часов) в первый и второй год обучения [1].

В вышеперечисленных учебных программах тема распознавания образов не упоминается. Машинное зрение в среде LEGO Mindstorms EV3 с использованием камеры Pixy рассматривается в **учебнике Овсяницкой Л.Ю.** В книге сформулировано современное видение понятия «машинное зрение», указаны особенности выбора версии камеры, совместимой с роботом EV3. Приведено большое количество примеров программ подсчета объектов, определения их цвета, размеров, положения, угла наклона, а также использования пропорционального управления для реализации наблюдения и следования роботом за движущимся цветным объектом и для движения по цветной линии. Учебник состоит из 6 глав:

1. Что такое машинное зрение.
2. Установка камеры и программного приложения PixyMon.
3. Обучение камеры цветам и цветным кодам.
4. Совместная работа камеры с роботом EV3.
5. Проекты с использованием камеры Pixy.
6. Где купить камеру Pixy [8].

Главным недостатком данного учебного пособия является то, что в нём рассматривается решение задач на распознавание образов только с использованием дорогостоящей камеры Pixy.

Несмотря на то, что распознавание образов стало актуальной исследовательской линией в области робототехники и компьютерного зрения, учебно-методическое обеспечение на эту тему отсутствует. Исходя из этого, было принято решение составить собственный лабораторный практикум.

При подробном рассмотрении программ курсов образовательной робототехники было замечено, что большинство из них предполагает следующий режим занятий: два академических часа в неделю. Такой режим занятий наиболее эффективен, так как за это время учащиеся могут выполнить задачу не только стандартным способом, но и предложить своё уникальное решение. Также было отмечено, что структура построения

образовательного процесса в большинстве учебных программах предполагает возрастающий уровень сложности в процессе освоения курса. Данные наблюдения были учтены при построении лабораторного практикума в рамках исследования.

Таким образом, в результате анализа различных робототехнических устройств, выбор был сделан в пользу самого распространенного в образовательных учреждениях конструктора – LEGO Mindstorms EV3, с которым совместимо множество веб-камер, обладающих различными характеристиками, подбор которых зависит от поставленных целей в процессе освоение машинного зрения.

К тому же, в ходе анализа графических и текстовых сред программирования было замечено, что программное обеспечение образовательной робототехники стремительно развивается, появляются среды, позволяющие интегрировать библиотеки машинного зрения. Для обучения решению задач по распознаванию образов, была выбрана среда программирования LeJOS.

Анализ учебных программ по робототехнике привёл к тому, что необходимо создание собственного лабораторного практикума для обучения решению задач по распознаванию образов, в связи с отсутствием разработок на данную тему.

ГЛАВА II. РАЗРАБОТКА ЛАБОРАТОРНОГО ПРАКТИКУМА ПО РОБОТОТЕХНИКЕ И МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ЕГО ИСПОЛЬЗОВАНИЮ

2.1. Структура и содержание лабораторного практикума

Разработанный в рамках исследования лабораторный практикум обеспечивает обучение старшеклассников решению задач по распознаванию образов в курсе робототехники. Данная разработка является продолжением лабораторного практикума по изучению невизуальной среды программирования leJOS в рамках исследовательской работы выпускника Уральского Государственного Педагогического Университета Герасимова Артема. Лабораторные работы предназначены для решения задач с использованием робототехнического комплекса LEGO Mindstorms EV3 в среде программирования leJOS на объектно-ориентированном языке Java.

Главная цель сборника лабораторных работ – познакомить учащихся с основными возможностями библиотек машинного зрения и научить применять полученные знания для решения базовых задач по распознаванию образов.

Для достижения поставленной цели был разработан практикум, включающий в себя 5 лабораторных работ. Структура каждой работы включает в себя:

1. Постановка цели работы.
2. Теоретические сведения, необходимые для выполнения работы.
3. Самостоятельные задания в рамках темы лабораторной работы.

Лабораторная работа 1. Захват экрана.

Цель работы: ознакомиться с библиотекой машинного зрения OpenCV и изучить основные этапы работы с камерой.

В ходе выполнения лабораторной работы, учащиеся знакомятся с основными пакетами библиотеки машинного зрения OpenCV. Результатом работы является простая демонстрационная программа, которая отображает

на экране модуля EV3 видео, полученное с веб-камеры. Для самостоятельной работы ученикам предлагается реализовать ручное управление порогом яркости изображения путем нажатия кнопок на модуле EV3.

Лабораторная работа 2. Распознавание цвета.

Цель работы: изучить технологию распознавания цвета.

В процессе выполнения лабораторной работы, учащимся предстоит проанализировать функцию получения проб цвета с камеры и на ее основе реализовать программу, в результате выполнения которой на дисплее робота будет выводиться сообщение о цвете, полученном с веб-камеры. Обработка изображения производится в цветовой модели HSV (Hue, Saturation, Value – цветовой тон, насыщенность, яркость). Необходимо объяснить учащимся назначение каждой характеристики представленной модели. Ученики должны прийти к выводу, что чем больше значения насыщенности и яркости, тем точнее определяется цвет.

Лабораторная работа 3. Распознавание геометрических фигур.

Цель работы: изучить технологию распознавания геометрических фигур.

В результате выполнения лабораторной работы на дисплее робота должно выводиться сообщение о форме объекта. Следует обратить внимание учеников на то, что в первую очередь происходит определение цвета объекта, после полученное изображение переводится в черно-белую картинку, где белыми обозначаются точки, цвет которых попал в заданный интервал, а чёрными – все остальные. Учащимся необходимо реализовать процедуру нахождения контура белой фигуры и вычисления ее координат, в соответствии с которыми и будет определяться, какой геометрической фигурой является объект.

Лабораторная работа 4. Слежение за объектом.

Цель работы: изучить технологию распознавания образа и реализовать слежение за объектом.

Результатом выполнения лабораторной работы является реализация программы слежения за объектом. В качестве объекта будет рассматриваться пятно определенного цвета. Учащиеся самостоятельно описывают процедуру распознавания цвета и поиск контура на основании знаний, полученных из предыдущих лабораторных работ. Затем им необходимо реализовать процедуру поиска центра и размера найденного контура объекта. Движение робота осуществить, в соответствии с тем, где этот контур расположен на картинке.

Лабораторная работа 5. Движение по линии.

Цель работы: реализовать метод движения робота по линии.

Перед выполнением лабораторной работы учащимся необходимо закрепить камеру так, чтобы в область захвата изображения попадала черная линия. В процессе выполнения работы учащимся предстоит реализовать уже знакомые им методы захвата изображения и определения черных объектов в выделенной области. Затем перед учениками ставится задача осуществить процедуру определения краев линии. В зависимости от смещения крайних точек к центру будет зависеть изменение скорости вращения моторов.

2.2. Методические рекомендации по использованию лабораторного практикума

2.2.1. *Подготовка оборудования к работе*

Перед использованием лабораторного практикума необходимо подготовить оборудование по следующему плану:

1. Подготовить SD-карту.
2. Установить компоненты leJOS EV3 на компьютер.
3. Создать SD-карту leJOS EV3.
4. Подключить модуль EV3 к компьютеру.
5. Установить Eclipse.
6. Подключить плагин leJOS EV3 в Eclipse.
7. Настроить подключение EV3 к компьютеру по USB или Bluetooth.

Требования к SD-карте:

- объем карты от 4Гб до 32Гб;
- файловая система – FAT32.

Для начала работы необходимо отформатировать SD-карту. После того, как подготовка SD-карты завершена, необходимо скачать последнюю версию leJOS EV3 с официального сайта проекта (lejos.org).

Во время установки необходимо выбрать JDK, рекомендуется использовать Java 7 (Рис. 7). Подходящую версию можно скачать с официального сайта *Oracle* (www.oracle.com).

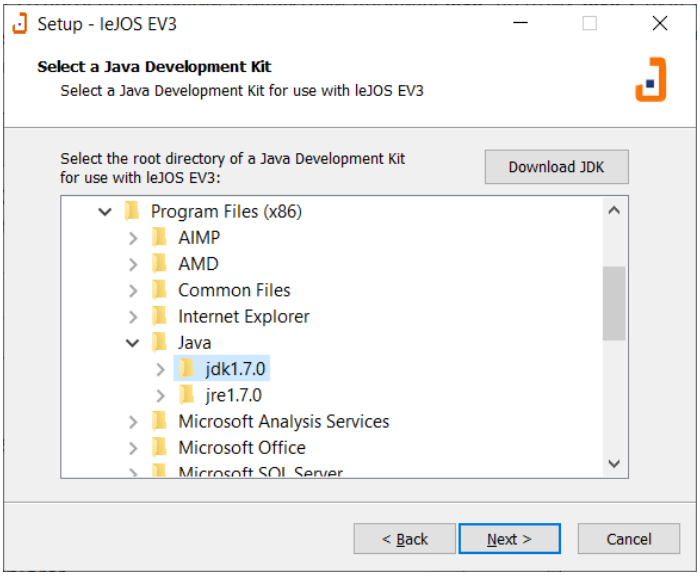


Рис. 7. Выбор JDK при установке leJOS EV3

На этапе выбора компонентов, рекомендуется установить все (Рис. 8).

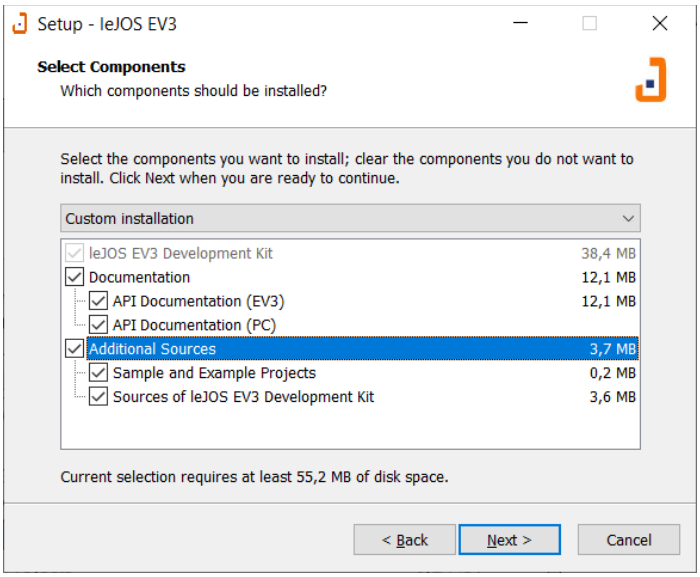


Рис. 8. Компоненты для установки leJOS

После завершения установки появится финальное окно, в котором можно запустить утилиту подготовки SD-карты, поставив галочку *Launch EV3SDCard utility*. В противном случае утилиту для записи SD-карты (Рис. 9) можно запустить вручную, открыв файл *ev3sdcard.bat*, расположенный в папке *bin*, внутри папки установленной программы.

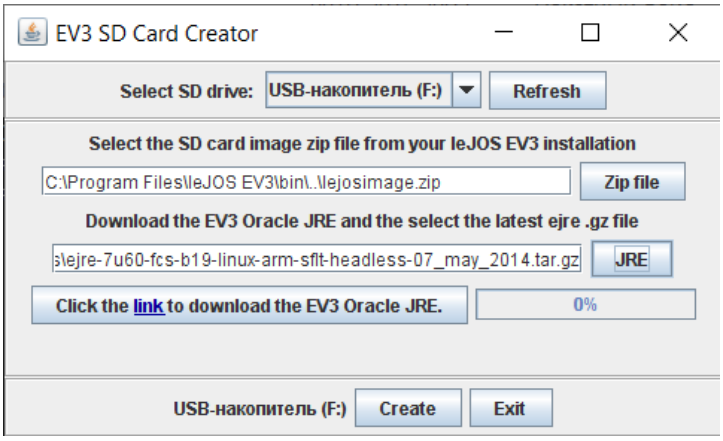
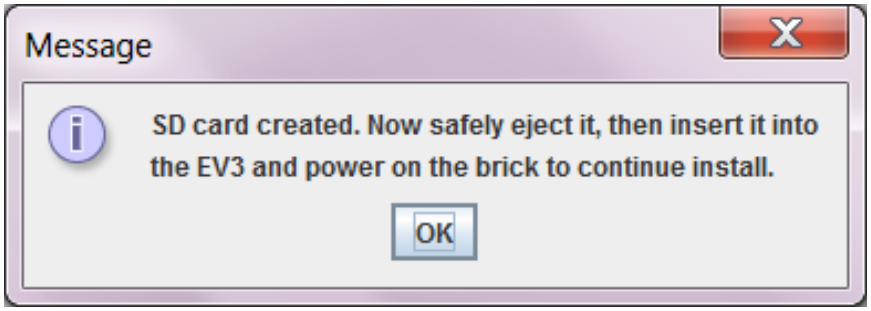


Рис. 9. Запись SD-карты

В верхнем поле указывается буква диска карты, в поле ниже – файл с образом leJOS (по умолчанию путь прописывается автоматически). Этот файл можно найти в папке, куда были установлены компоненты leJOS EV3. В самом нижнем поле необходимо выбрать файл со средой выполнения Java, который нужно предварительно скачать с официального сайта *Oracle* (для скачивания потребуется регистрация на сайте).

После заполнения всех полей необходимо нажать на кнопку *Create*. На экране должно появиться подобное сообщение:



После безопасного извлечения карты необходимо вставить ее в выключенный микроконтроллер EV3 и включите его, нажав центральную кнопку на блоке. В конце процесса установки модуль EV3 перегрузится и на экране появится меню leJOS EV3 (Рис. 10).

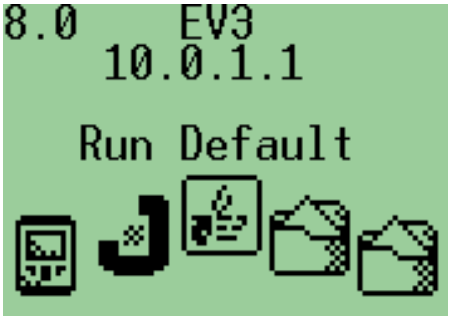


Рис. 10. Меню leJOS EV3

Подключить модуль EV3 к компьютеру можно двумя способами: через Bluetooth или USB.

Для подключения через Bluetooth необходимо настроить соединение между модулем EV3 и компьютером, а затем создать личную сеть Bluetooth (PAN).

Для начала требуется включить видимость обоих устройств. В модуле EV3 данная настройка находится в пункте главного меню *Bluetooth*. Надпись *Visibility on* означает, что видимость включена, в противном случае включить ее можно, дважды нажав на центральную кнопку модуля EV3. Видимость компьютера настраивается в меню *Bluetooth.*

После того как модуль EV3 и компьютер подключены друг к другу настраивается сеть PAN. Для этого необходимо открыть *Панель управления → Сеть и Интернет → Центр управления сетями и общим доступом* и щёлкнуть по пункту *Изменение параметров адаптера* на левой панели окна. В открывшемся окне нужно выделить иконку *Сетевое подключение Bluetooth* с красным крестиком, после щелкнуть по кнопке *Просмотр сетевых устройств Bluetooth*. В открывшемся окне необходимо выбрать устройство EV3 и щёлкнуть по пункту меню *Подключаться через → Точка доступа* (Рис. 11).

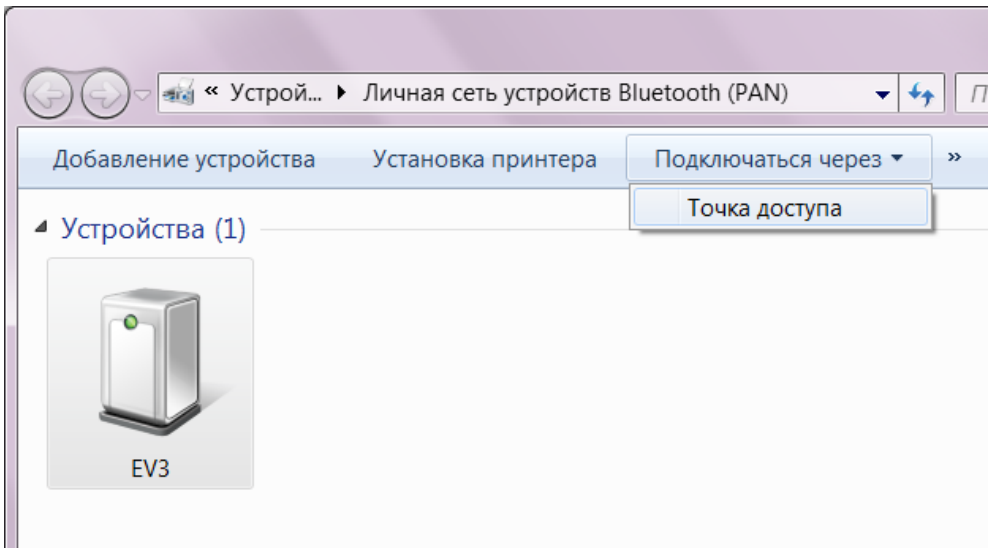


Рис. 11. Настройка точки доступа

Для того, чтобы настроить оборудование для соединения по USB, требуется подключить к компьютеру включенный модуль EV3 с вставленной в него картой, подготовленной ранее. После подключения в окне *диспетчера устройств* появится неизвестное оборудование (Рис. 12).

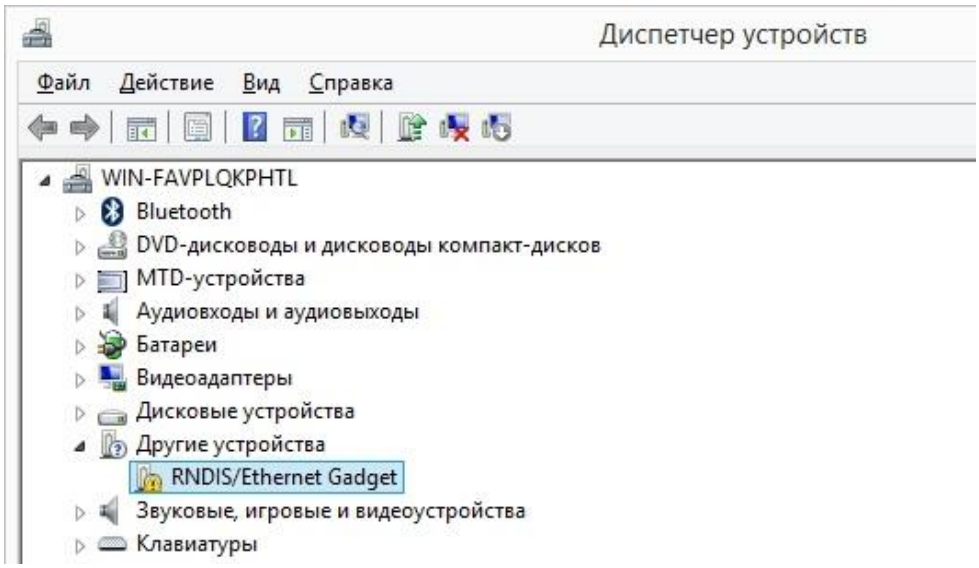


Рис. 12. Неопознанный блок EV3

Для настройки устройства необходимо щелкнуть правой клавишей мыши на неопознанном устройстве и выбрать в контекстном меню *Обновить драйвер*. В появившемся окне производится выбор действие *Выполнить поиск драйверов на этом компьютере* (Рис. 13).

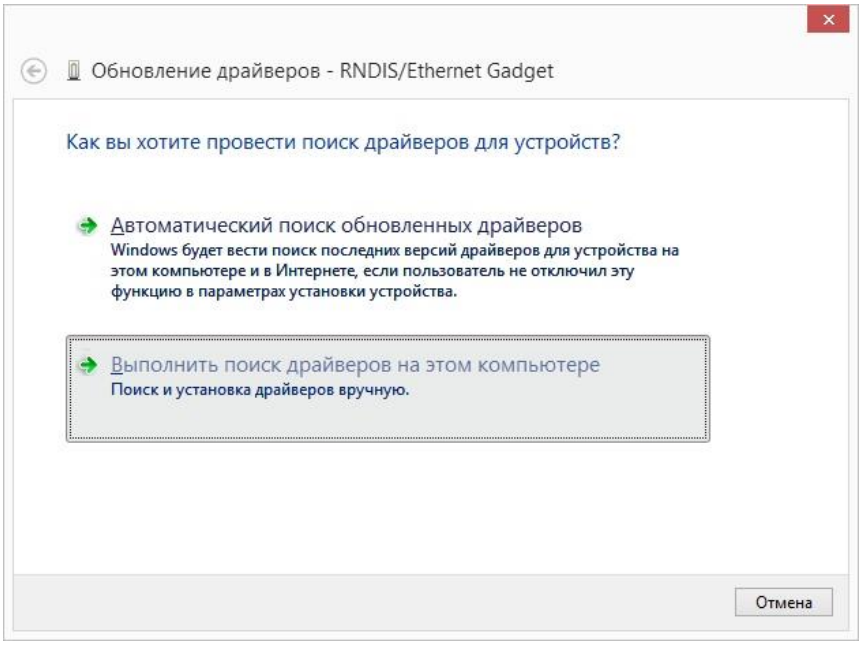


Рис. 13. Выбор типа поиска

В новом окне необходимо щелкнуть по пункту *Выбрать драйвер из списка уже установленных драйверов*. На следующем этапе выбирается пункт *Сетевые адаптеры* (Рис. 14).

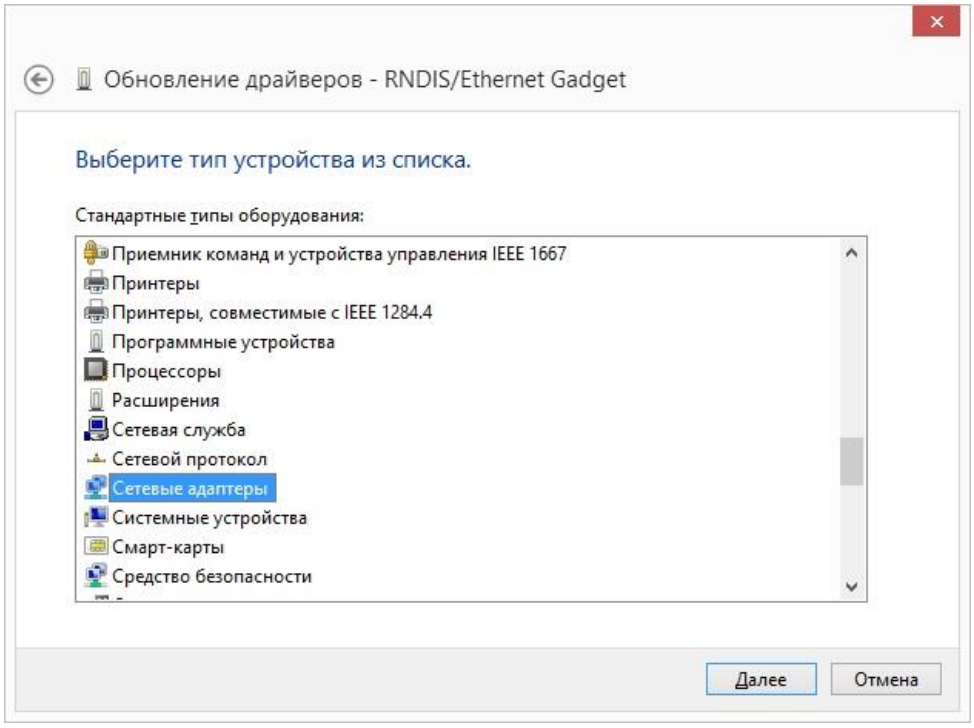


Рис. 14. Выбор типа устройства

Из списка *Изготовитель* необходимо выбрать *Microsoft Corporation* и подходящий сетевой адаптер (Рис. 15).

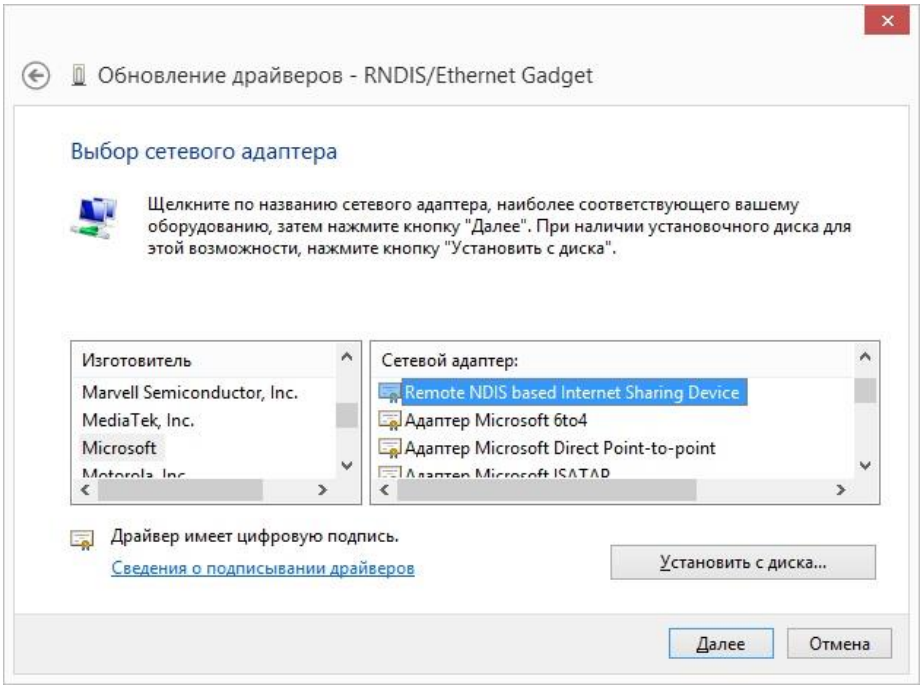


Рис. 15. Выбор сетевого адаптера

Для установки среды разработки Eclipse необходимо скачать с официального сайта архив с любой версией программного продукта и распаковать его на компьютере в том же диске, где установлена Java. В процессе запуска Eclipse запросит место расположения рабочей папки.

После запуска Eclipse выбирается тот же JDK, который был выбран при установке leJOS на компьютер. Для этого необходимо щелкнуть по пункту меню *Window* → *Preferences*, после в диалоге настроек выбрать *Java* → *Installed JREs,* нажать на кнопку *Search...* и выбрать папку с JDK. После поиска в списке *Installed JREs* появится найденная виртуальная машина, необходимо поставить галочку рядом с ней (Рис. 16).

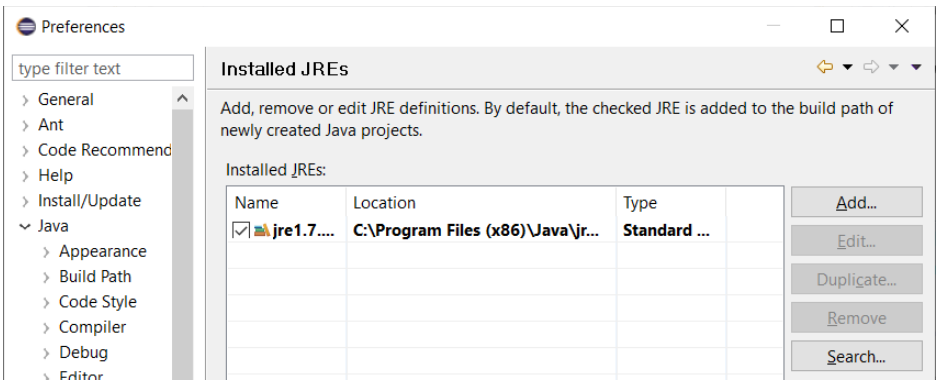


Рис. 16. Выбор JDK в Eclipse

Теперь необходимо установить плагин leJOS EV3 в Eclipse. Для этого нужно вызвать пункт меню *Help* → *Eclipse Marketplace...,* в появившемся

диалоге выбрать *leJOS EV3* и нажать кнопку *Install* напротив плагина *leJOS EV3 Plug-In* (Рис. 17).

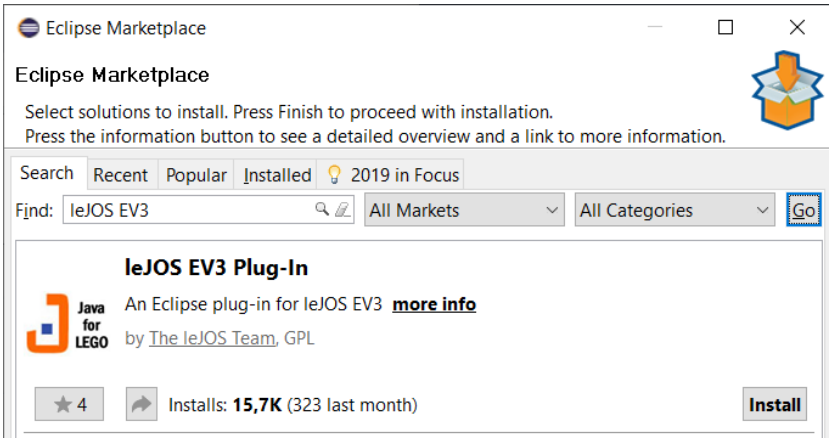


Рис. 17. Магазин Eclipse

На следующем шаге необходимо принять лицензионное соглашение (Рис. 18).

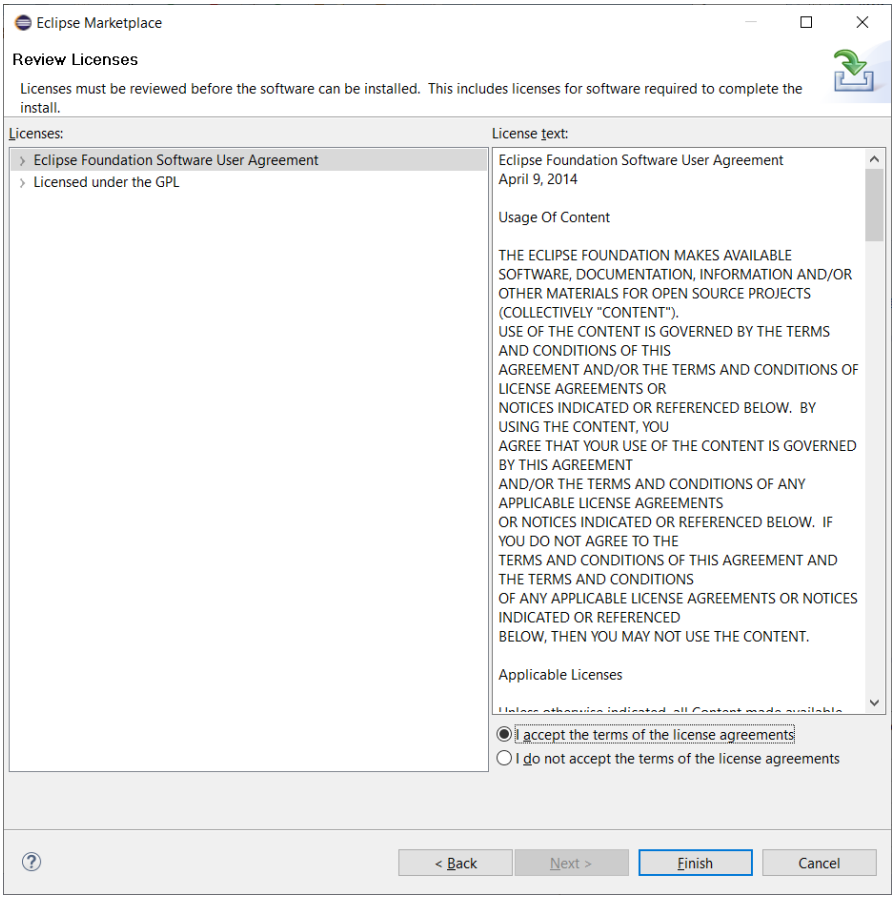
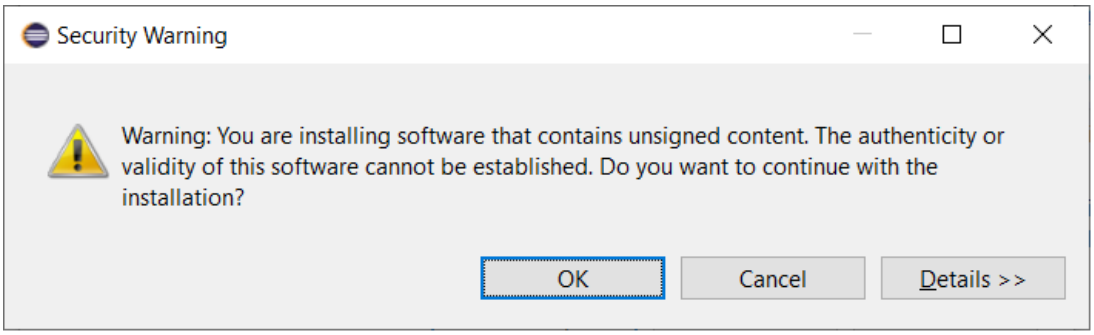
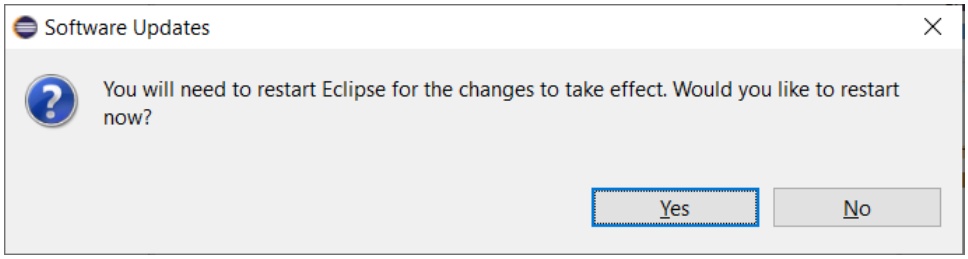


Рис. 18. Пользовательское соглашение на установку плагина leJOS EV3

После этого при установке плагина необходимо согласиться с появившемся предупреждением о том, что устанавливаемое ПО не имеет подписи:



После завершения процесса установки появится сообщение с предложением перезагрузки Eclipse:



После перезапуска необходимо проверить настройки плагина, для этого нужно вызвать пункт меню *Window* → *Preferences*, в появившемся окне настроек щелкнуть в списке слева по *leJOS EV3*. В поле *EV3_HOME* должен быть указан путь к папке, в которой установлен *leJOS EV3*. Напротив пункта *Run Tools in separate JVM* должна быть установлена галочка, а напротив *Use ssh and scp* – нет. Здесь же необходимо поставить галочку *Connect to named brick* и в поле *Name* указать IP-адрес используемого модуля EV3, который можно посмотреть в главном меню микроконтроллера. После проверки и настройки окно можно закрыть нажатием клавиши *OK* (Рис. 19) [12].

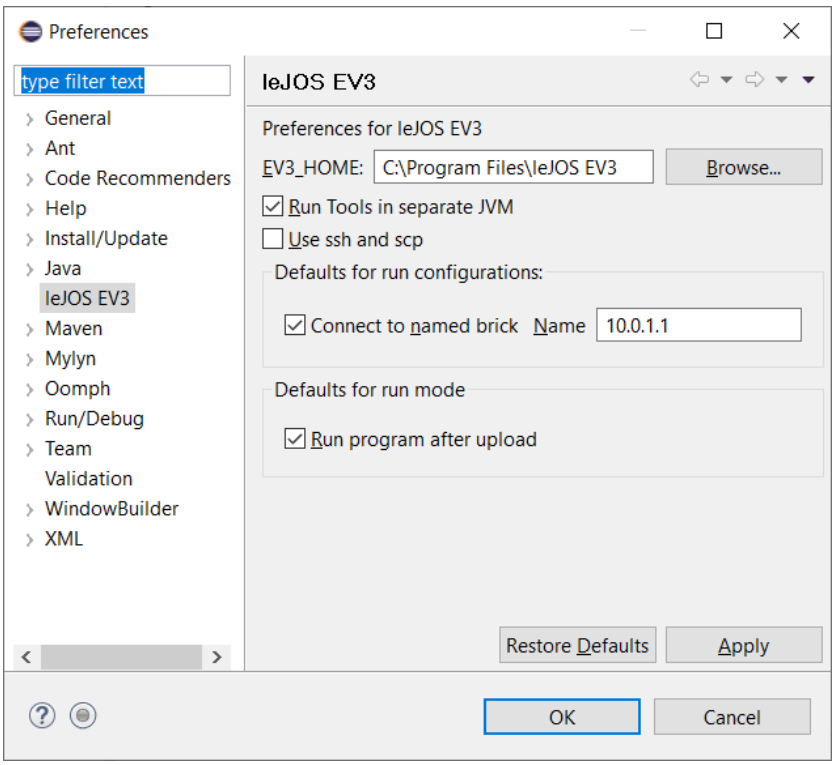


Рис. 19. Настройка плагина

2.2.2. *Методические рекомендации по использованию лабораторного практикума*

Разработанный практикум можно использовать для обучения учеников 9-11 классов средних общеобразовательных учреждений, предварительно прошедших курс по программированию в невизуальной среде leJOS на языке Java.

После получения необходимого представления о программировании робота на объектно-ориентированном языке, предполагается самостоятельное выполнение лабораторных работ по распознаванию образов под руководством преподавателя. В процессе занятия учитель наблюдает за ходом выполнения работы и консультирует учащихся по появляющимся вопросам. При столкновении с проблемами, возникающими у большинства учеников, рекомендуется их совместное решение. Практикум рассчитан на 7 академических часов: по одному часу на каждую лабораторную работу, за исключением четвертой работы, посвященной слежению за объектом, на ее выполнение отводится 3 часа.

При работе с камерой требуется соблюдать определенные правила и последовательности действий. Данные особенности работы рассматриваются уже при выполнении первой лабораторной работы, которая посвящена теме «Захват экрана».

Работа с камерой заключается в выполнении следующих шагов:

1. Получение экземпляра объекта Video.
2. Подготовка камеры к работе (вызов функции open).
3. Создание массива байтов для хранения кадра (вызов функции createFrame).
4. Получение кадра (вызов функции grabFrame).

Первые три шага выполняются один раз, а четвертый шаг воспроизводится каждый раз, когда необходимо захватить изображение с камеры. При выполнении задания, учащиеся пришли к выводу, что для получения видео последний шаг должен повторяться в цикле. Для работы с изображением используется формат YUYV. Данный формат отделяет информацию полученного цвета от яркости, с которой рациональнее работать, используя пороговые значения.

При выводе изображения на экран микроконтроллера необходимо обратить внимание на то, что программа преобразует полученное изображение в чёрно-белое, поскольку дисплей EV3 монохромный. Преобразование происходит следующим образом: точки темнее установленного порога преобразуются в чёрные, остальные – в белые. Так же следует обратить внимание, что для хранения порога яркости заводится целочисленная переменная threshold.

При выполнении первой лабораторной работы, ученикам предлагается самостоятельно реализовать ручное управление яркостью изображения путем нажатия кнопок на модуле микроконтроллера EV3. Для выполнения поставленной рекомендуется использовать условный оператор *if*.

Во второй лабораторной работе ученикам предлагается изучить и проанализировать функцию получения проб цвета с камеры. Данная функция

возвращает информацию о цветовом тоне, насыщенности и яркости объекта. Именно цветовой тон, интервал которого изменяется от 0 до 179, определяет цвет объекта. Путем сравнения полученного значения с выделенными учениками цветовыми интервалами осуществляется процесс определения цвета. Насыщенность и яркость находятся в пределах от 0 до 255. Чем меньше насыщенность, тем ближе к белому или серому будет цвет, чем меньше яркость – тем ближе к чёрному. Следует обратить внимание учеников на то, что наибольшие значения насыщенности и яркости позволяют упростить процесс определения цвета. При выполнении задания, учащиеся должны прийти к выводу, что для непрерывного определения цвета объектов фрагмент кода, отвечающий за сравнение цветового тона, должен повторяться в цикле. В качестве объекта можно использовать карточки из цветного картона или пластмассовое фигуры разного цвета. Рекомендуется использовать такие цвета, как красный, синий, желтый и зеленый.

Основой третьей лабораторной работы является программный код для определения цвета, полученный учениками в ходе предыдущей работы. Следующим шагом является перевод полученного изображения в черно-белое, где белыми обозначаются точки, цвет которых попал в заданный интервал, а чёрными – все остальные. Учащимся необходимо реализовать процедуру нахождения контура белой фигуры и вычисления ее координат, в соответствии с которыми и будет определяться, какой геометрической фигурой является объект. Процесс вычисления координат фигуры и их последующая обработка может стать достаточно сложной задачей для учеников, которую следует вынести на совместное обсуждение и найти таким образом общее решение. Для выполнения данной работы необходимо подготовить объекты треугольной, квадратной и круглой формы. Фигуры могут быть плоскими (вырезанными из цветного картона), либо объемными (цветные пластиковые фигуры).

Выполнение четвертой лабораторной работы следует разделить на три этапа, для каждого из которых необходимо выделить отдельное занятие. В

основе данной работы в очередной раз лежит определение цвета, так как в качестве объекта будет рассматриваться цветное пятно, однако на этот раз данное действие происходит однократно в начале программы. На первом занятии учащимся предстоит самостоятельно описать процедуру распознавания цвета и поиск контура на основании знаний, полученных из предыдущих лабораторных работ. На этом же уроке учащимся необходимо реализовать процедуру поиска центра и размера найденного контура объекта. В качестве объекта рекомендуется использовать пластиковый шарик красного или синего цвета, так как это наиболее яркие и насыщенные цвета. На втором занятии рекомендуется осуществить слежение за объектом путем манипулирования непосредственно самой камеры, которую предварительно следует закрепить на подвижном механизме. Данная конструкция может быть присоединена к сервомотору несколькими способами: путем использования различных механических передач либо напрямую к его вращающейся части. На третьем занятии реализуется движение робота вперед-назад в зависимости от изменений размера контура объекта, полученного с изображения камеры. Также на этом занятии осуществляется поворот робота на месте влево и вправо в соответствии расположения контура объекта относительно центра кадра.

Перед выполнением пятой лабораторной работы учащимся необходимо закрепить камеру так, чтобы в область захвата изображения попадала черная линия. Также следует подготовить поле, состоящее из непрерывной линии. Для усложнения задачи рекомендуется использовать поле с перекрестками и прямыми углами. В процессе выполнения работы учащимся предстоит реализовать уже знакомые им методы захвата изображения и определения черных объектов в выделенной области. Последующий процесс рулевого управления роботом в зависимости от положения черной линии в выделенной области можно реализовать несколькими способами, в основе которых лежит метод определения краев линии. В зависимости от смещения крайних точек к центру будет зависеть изменение направления вектора

движения. Так как лабораторная работа является не только завершающей в данном практикуме, но и достаточно сложной для выполнения, углубленное изучение поставленной задачи может стать простором для нахождения ее уникального решения.

2.3. Апробация лабораторного практикума

Апробация материалов исследования проводилась в 2019 году в Частном Общеобразовательном Учреждении Средняя Общеобразовательная Школа «Творчество», в которой принимала участие группа учеников 10 класса. Занятия проводились один раз в неделю по два академических часа, лабораторные работы выполнялись в установленном порядке.

Перед началом исследования, учащиеся прошли обучение по лабораторному практикуму «Программирование в невизуальной среде LeJOS на языке Java», автором которого является выпускник Уральского Государственного Педагогического Университета Герасимов А.А. Получив необходимое представление о программировании робота на объектно-ориентированном языке, ученики преступили к решению задач по распознаванию образов.

Перед выполнением каждой лабораторной работы учащиеся коллективно обсуждали содержание программы и только после этого индивидуально решали поставленную перед ними задачу. При возникновении трудностей ученики обращались к преподавателю и общими усилиями выполняли задание.

При выполнении первой лабораторной работы, ученикам было предложено самостоятельно реализовать ручное управление яркостью изображения путем нажатия кнопок на модуле микроконтроллера EV3. Для выполнения поставленной задачи учащиеся использовали условный оператор if. В результате ученики пришли к написанию следующего кода (Рис. 20). Некоторые ученики использовали логическую переменную, отвечающую за автоматическую настройку яркости, что увеличивало избыточность кода, но не нарушало общую логику программы. В основном все учащиеся работали в

одном темпе, структура и содержание задания не вызывало у них затруднений. В конечном итоге все ученики успешно справились с первой лабораторной работой.

```
import lejos.hardware.video.Video;
import lejos.hardware.video.YUYVImage;
import java.io.IOException;
import lejos.hardware.BrickFinder;
import lejos.hardware.Button;
import lejos.hardware.lcd.GraphicsLCD;
 public class CameraDemo {
     public static void main(String[] args) throws IOException{
             Video video = BrickFinder.getDefault().getVideo();
             video.open(160,120);
             byte[] frame = video.createFrame();
             YUYVImage image = new YUYVImage(frame, video.getWidth(), video.getHeight());
             GraphicsLCD graphics = BrickFinder.getDefault().getGraphicsLCD();
             int threshold = 128;
             while (!Button.ESCAPE.isDown()) {
                 video.grabFrame(frame);
                 image.display(graphics, 0, 0, threshold);
                 if (Button.UP.isDown())
                     if (threshold < 255)
                         threshold++;
                 if (Button.DOWN.isDown())
                     if (threshold < 0)
                         threshold--;
                 if (Button.ENTER.isDown())
                     threshold = image.getMeanY();
}}}
```

Рис. 20. Программа Захват экрана

В ходе выполнения второй лабораторной работы ученикам удалось написать программу по определению четырёх цветов: зелёный, жёлтый, красный и синий. Для сравнения цветового тона, полученного из функции определения проб цвета с камеры, с цветовым интервалом ученики использовали условный оператор if. Процесс определения цвета учащиеся организовали с использованием цикла while. Небольшие затруднения у школьников вызвало составление условия: изначально ученики неправильно определили интервал, но путём проведения экспериментов им удалось отладить программу. В основном все учащиеся работали в одном темпе, структура и содержание задания не вызывало у них затруднений. В конечном итоге все ученики успешно справились со второй лабораторной работой.

Третья лабораторная работа по распознаванию геометрической формы объекта вызвало у школьников наибольшие затруднения. На её выполнения потребовалось большее количество занятий, чем планировалось. Трудности у учащихся возникли при реализации процедуры нахождения контура фигуры и вычисления её координат. Решение данного вопроса было вынесено на

коллективное обсуждение, и дальнейший процесс написания программы происходил совместно под руководством преподавателя. В конечном итоге ученики пришли к выводу, что фигуру можно определить по количеству найденных у контура вершин. По итогам апробации данной лабораторной работы было принято решение поместить ее в конец практикума, так как она обладает наибольшим уровнем сложности, также время на её выполнение было увеличено до четырёх академических часов.

Выполнение четвертой лабораторной работы стало возможностью для реализации творческих идей учеников, которые создавали уникальные подвижные конструкции для закрепления камеры. Перед конструированием учащиеся уже достаточно уверено писали программу для реализации слежения за объектом. Все школьники были увлечены процессом, активно участвовали в обсуждениях и предлагали разные решения. Темп выполнения заданий отличался, но преуспевающие ученики помогали отстающим, чтобы поддерживать равномерный ритм работы. В итоге все ученики успешно справились с данной лабораторной работой.

Перед тем, как приступить к выполнению пятой лабораторной работы, учащиеся закрепили камеру параллельно поверхности. Конструкции у всех получились разные: некоторые ученики расположили камеру максимально низко, а кто-то, напротив, закрепил устройство очень высоко. В процессе отладки программы некоторые учащиеся изменяли высоту расположения камеры. С реализацией основной идеи программы – определения краев линии и центра изображения – проблем у школьников не возникло. Процесс рулевого управления роботом в зависимости от положения черной линии в выделенной области, учащиеся реализовывали при помощи разного инструментария среды разработки. После того, как все школьники справились со стандартной задачей, им было предложено поле с перекрестками и прямыми углами. Для прохождения усложненной траектории ученики регулировали скорость, добавляли условия и т.д. Преуспевающие ученики реализовали изменение скорости путем

добавления/вычитания к ней коэффициента, умноженного на удаление от центральной точки.

В результате освоения лабораторного практикума, учащиеся научились:

- использовать различные библиотеки, в том числе библиотеку машинного зрения OpenCV;
- программировать на объектно-ориентированном языке Java;
- решать задачи по распознаванию образов;
- закреплять на робототехническом устройстве веб-камеру.

Результат апробации разработанного курса показал, что наибольшее затруднение у учащихся вызвала работа с новой библиотекой OpenCV, на изучение которой в ходе обучения было выделено большее количество времени. В целом все ученики успешно справились с поставленными целями и задачами курса.

В завершении прохождения курса с учащимися была проведена беседа, в ходе которой было выявлено, что всем участникам процесса понравилась тема распознавания образов, изучением которой школьники хотели бы продолжить заниматься в следующем учебном году.

ЗАКЛЮЧЕНИЕ

В ходе исследования были изучены популярные на сегодняшний день робототехнические конструкторы и устройства захвата изображения. В частности, проведено сравнение аппаратных средств с целью выбора наиболее бюджетного и оптимального аппаратного обеспечения для решения задач по распознаванию образов.

Рассмотрены среды программирования для разработки программ с использованием библиотек машинного зрения. В ходе анализа программного обеспечения, были выделены особенности и возможности сред разработки для реализации задач по распознаванию образов.

В процессе анализа методических разработок в области образовательной робототехники была определена структура и режим занятий, а также выявлена потребность в создании собственного лабораторного практикума для решения задач по распознаванию образов.

В соответствии результатам исследования был разработан лабораторный практикум для решения задач по распознаванию образов, который включает в себя следующие темы:

1. Захват экрана.
2. Определение цвета.
3. Определение геометрической формы объекта.
4. Слежение за объектом.
5. Следование по линии.

К данному лабораторному практикуму были разработаны и сформулированы методические рекомендации по его использованию.

Результаты апробации подтвердили заинтересованность школьников в изучении данной темы.

В процессе работы над темой исследования выявлено недостаточное количество необходимой литературы, поэтому было решено обратиться к сообществу, которое способствовало решению данной проблемы. Хотелось бы выразить отдельную благодарность Лыткину Сергею Дмитриевичу,

руководителю Центра электронного обучения Федерального государственного автономного образовательного учреждения высшего образования Северо-Восточного федерального университета им. М.К. Аммосова (г. Якутск, республика Саха) и Безручко Фёдору Владимировичу, директору Центра робототехники Тихоокеанского государственного университета (г. Хабаровск, Хабаровский край).

По результатам данного исследования была опубликована статья «Методика обучения учащихся старших классов решению задач по распознаванию образов в курсе робототехники» в межвузовском сборнике «Информационно-коммуникационные технологии в образовании» [2].

Таким образом, следует считать, что задачи работы полностью выполнены и цель исследования достигнута.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Васильев А.Д. Образовательная программа дополнительного образования «Робототехника». СПб.: ФМЛ №422, 2011.
2. Димитрова М.Д., Шимов И.В. Методика обучения учащихся старших классов решению задач по распознаванию образов в курсе робототехники // Актуальные вопросы преподавания математики, информатики и информационных технологий: межвуз. сб. науч. работ. Екатеринбург: Изд-во УрГПУ, 2019. № 4. С. 227-233.
3. Димитрова М.Д., Шимов И.В. Организация самостоятельной работы школьников в процессе обучения основам курса робототехники //Педагогическое образование в России. 2018, № 3. С. 224-230
4. Копосов Д.Г. УМК для средней школы «Первый шаг в робототехнику», 2012.
5. Литвинов Ю.В. Визуальные средства программирования роботов и их использование в школах // Современные информационные технологии и ИТ-образование. М.: 2012. С. 858-868.
6. Литвинов Ю.В., Кириленко Я.А. TRIK Studio: среда обучения программированию с применением роботов // V Всероссийская конференция «Современное технологическое обучение: от компьютера к роботу». 2015. С. 5-7.
7. Мордвинов Д.А., Литвинов Ю.В. Сравнение образовательных сред визуального программирования роботов. // Компьютерные инструменты в образовании. СПб: Компьютер в учебном процессе, 2016. С. 145-161.
8. Овсяницкая, Л.Ю. Машинное зрение в среде Lego Mindstorms EV3 с использованием камеры Pixy (CMUcam5). Электронная книга, 2016. 168 с.
9. Опалёва Д.А. Языки программирования и методы трансляции. СПб.: БХВ-Петербург, 2005. 480 с.

10. Попова Т.Г. Образовательная робототехника: дайджест актуальных материалов. Екатеринбург: ГАОУ ДПО СО «Институт развития образования», 2015. 70 с.
11. Предко М. 123 эксперимента по робототехнике. М.: НТ Пресс, 2007. 544 с.
12. Программирование робота LEGO Mindstorms EV3 на Java // ПрогХаус URL: http://www.proghouse.ru/article-box/55-lejos-ev3 (дата обращения: 20.01.2019).
13. Русская версия EV3 Basic // Строим вместе с Карандашом и Самоделкиным URL: http://karandashsamodelkin.blogspot.com/2015/11/ev3-basic.html (дата обращения: 24.04.2019).
14. Федеральный государственный образовательный стандарт основного общего образования. URL: https://fgos.ru/ (дата обращения: 21.02.2019).
15. Филиппов С.А. Дополнительная образовательная программа «Робототехника: конструирование и программирование». СПб.: ФМЛ №239, 2011.
16. Шимов И.В. Применение робототехнических устройств в обучении программированию школьников // Педагогическое образование в России. 2013. № 1. С. 185-188.
17. AutoNet // Робототехника. Инженерно-технические кадры инновационной России URL: http://russianrobotics.ru/competition/autonet/autonet-14/ (дата обращения: 12.02.2019).
18. Bagnall B. Maximum Lego Mindstorms EV3: Building Robots With Java Brains // Variant Press, 2014. С. 102-105.
19. EV3 программирование // Робототехника для начинающих URL: https://legoteacher.ru/robototexnika-dlya-nachinayushhix/ev3-programmirovanie.html (дата обращения: 03.05.2019).
20. Ev3dev Home URL: www.ev3dev.org (дата обращения: 05.05.2019).
21. Fischertechnik // URL: http://fischertechnik.ru/ (дата обращения: 18.03.2019).

22. Jost B., Ketterl M., Budde R. Graphical Programming Environments for Educational Robots: Open Roberta-Yet Another One // Multimedia (ISM). 2014. C. 381-386.
23. LEGO Education // URL: legoeducation.com/mindstorms (дата обращения: 20.03.2019).
24. LeJOS // Java for Lego Mindstorms URL: http://www.lejos.org/ (дата обращения: 24.03.2019).
25. Papert S. Mindstorms: Children, Computers, and Powerful Ideas. New York, USA: Basic Books, 1980. 242 с.
26. PIXYCAM // URL: https://pixycam.com/ (дата обращения: 20.03.2019).
27. Resnick M., Maloney J., Monroy-Hernandez A. Scratch: programming for all // Communications of the ACM. 2009. C. 60-67.
28. RobotC // Сайт Паяльник URL: http://cxem.net/software/robotc.php (дата обращения: 24.04.2019).
29. Stefanovic M., Cvijetkovic V., Matijevic M. A LabVIEW-based remote laboratory experiments for control engineering education // Computer Applications in Engineering Education. Kragujevac, Serbia: Wiley InterScienc, 2011. C. 538-549.
30. TRIK // URL: https://trikset.com/ (дата обращения: 18.03.2019).
31. Vision Subsystem // URL: http://www.mindsensors.com/vision-for-robots/191-vision-subsystem-v5-for-nxt-or-ev3-with-fixed-lens (дата обращения: 18.03.2019)

ПРИЛОЖЕНИЕ

Лабораторная работа №1

Захват экрана

Цель работы: ознакомиться с библиотекой машинного зрения OpenCV и изучить основные этапы работы с камерой.

Для предотвращения непредвиденных ситуаций и возникновения ошибок, воспользуемся *обработчиком событий*, с его назначением и принципом работы вы познакомитесь при углубленном изучении объектно-ориентированного языка программирования Java.

Добавим исключение *IOException* в метод *main.*

```
public static void main(String[] args) throws IOException{}
```

Импортируем следующие пакеты:

```
import java.io.IOException;
import lejos.hardware.video.Video;
import lejos.hardware.video.YUYVImage;
import lejos.hardware.BrickFinder;
import lejos.hardware.Button;
import lejos.hardware.lcd.GraphicsLCD;
```

Работа с камерой заключается в выполнении следующих шагов:

1. *Получение экземпляра объекта.*

```
Video video = BrickFinder.getDefault().getVideo();
```

2. *Подготовка камеры к работе.*

```
video.open(160,120);
```

3. *Создание массива байтов для хранения кадра.*

```
byte[] frame = video.createFrame();
```

4. *Получение кадра.*

```
video.grabFrame(frame);
```

Первые три шага выполняются один раз, а четвертый шаг воспроизводится каждый раз, когда необходимо захватить изображение с камеры.

Для работы с изображением используется формат ***YUYV***. Данный формат отделяет информацию полученного цвета от яркости, с которой рациональнее работать, используя пороговые значения.

```
YUYVImage image = new YUYVImage(frame, video.getWidth(), video.getHeight());
GraphicsLCD graphics = BrickFinder.getDefault().getGraphicsLCD();
```

При выводе изображения на экран микроконтроллера программа преобразует полученное изображение в чёрно-белое, поскольку дисплей EV3 монохромный. Преобразование происходит следующим образом: точки темнее установленного порога преобразуются в чёрные, остальные – в белые.

Для хранения порога яркости необходимо создать *целочисленную переменную* ***light***.

Для передачи полученного изображения с веб-камеры на дисплей микроконтроллера EV3 воспользуемся методом *display*.

```
image.display(graphics, 0, 0, light);
```

Задания для самостоятельной работы:

1. Реализуйте автоматическую коррекцию яркости изображения.
2. Реализуйте ручную настройку яркости изображения путем нажатия на кнопки модуля микроконтроллера EV3.

Лабораторная работа №2

Распознавание цвета

Цель работы: изучить технологию распознавания цвета.

Для начала работы импортируем следующие пакеты:

```
import org.opencv.core.Core;
import org.opencv.core.Mat;
import org.opencv.core.Rect;
import org.opencv.core.Size;
```

```
import org.opencv.highgui.Highgui;
import org.opencv.highgui.VideoCapture;
import org.opencv.imgproc.Imgproc;
```

Загрузим библиотеку OpenCV и создадим вспомогательный класс для захвата видео.

```
System.loadLibrary(Core.NATIVE_LIBRARY_NAME);
VideoCapture videoCapture = new VideoCapture(0);
```

Зададим ширину и высоту кадра.

```
videoCapture.set(Highgui.CV_CAP_PROP_FRAME_WIDTH, 160);
videoCapture.set(Highgui.CV_CAP_PROP_FRAME_HEIGHT, 120);
```

Считаем изображение с камеры и запишем полученное значение в переменную ***image***.

```
Mat image = new Mat();
videoCapture.read(image);
```

Запишем пробу цвета в центре кадра в одномерный массив ***sampleColor***.

```
double[] sampleColor = getSampleColor(videoCapture);
```

Обработка изображения производится в цветовой модели HSV (Hue, Saturation, Value – цветовой тон, насыщенность, яркость). Цветовой тон как раз и определяет цвет. В OpenCV он может принимать значения от 0 до 179 (Рис. 21), а насыщенность и яркость – от 0 до 255. Чем меньше насыщенность, тем ближе к белому или серому будет цвет, чем меньше яркость – тем ближе к чёрному.

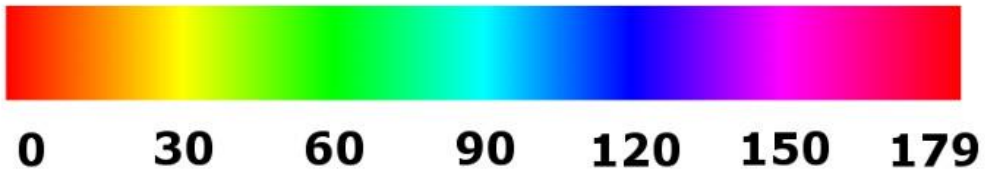


Рис. 21. Диапазон цветовых значений в библиотеке OpenCV

Создадим *вещественную* переменную, в которой будет храниться значение ***цветового тона***, полученное из массива ***sampleColor.***

Проанализируйте функцию ***getSampleColor()***, которая получает пробы цвета с камеры

```
private static double[] getSampleColor(VideoCapture videoCapture) {
  Mat image = new Mat();
  Size ksize = new Size(5, 5);
  double h = 0, s = 0, v = 0;
  int sampleCount = 7; //количество проб
  //увеличиваем счетчик только при выполнении условия
  for (int i = 0; i < sampleCount; ) {
    if (videoCapture.read(image) && !image.empty()) {
      Rect snippetROI = new Rect(image.width()/2 - 5,image.height()/2 - 5, 10, 10);
      Mat snippet = image.submat(snippetROI);
      Imgproc.blur(snippet, snippet, ksize);
      Imgproc.cvtColor(snippet, snippet, Imgproc.COLOR_BGR2HSV);
      double[] color = snippet.get(5, 5);
      h += color[0];
      s += color[1];
      v += color[2];
      i++;
      Delay.msDelay(100);
    }
  }
  return new double[] {
    Math.round(h / sampleCount),
    Math.round(s / sampleCount),
    Math.round(v / sampleCount)
  };
}
```

В данном случае проба цвета берётся 7 раз с небольшим интервалом времени, и конечный цвет вычисляется как среднее арифметическое от взятых проб. Данное действие позволяет снизить зависимость от мерцаний и подстройки баланса белого.

Задание для самостоятельной работы: реализуйте программу определения цвета объекта (желтый, зеленый, синий, красный) по его цветовому тону.

Лабораторная работа №3

Распознавание геометрических фигур

Цель работы: изучить технологию распознавания геометрических фигур.

Для начала работы импортируем следующие пакеты:

```
import java.io.ByteArrayInputStream;
import java.net.URL;
import java.util.ResourceBundle;
import java.util.*;
import java.util.concurrent.Executors;
import java.util.concurrent.ScheduledExecutorService;
import java.util.concurrent.TimeUnit;
import org.opencv.core.*;
import org.opencv.core.CvType.CV_32S;
import org.opencv.core.CvType.CV_64F;
import org.opencv.imgproc.*;
import org.opencv.highgui.*;
import org.opencv.imgproc.Imgproc.ADAPTIVE_THRESH_GAUSSIAN_C;
import org.opencv.imgproc.Imgproc.BORDER_DEFAULT;
import org.opencv.imgproc.Imgproc.COLOR_BGR2GRAY;
import org.opencv.imgproc.Imgproc.CV_SHAPE_RECT;
import org.opencv.imgproc.Imgproc.MORPH_CLOSE;
import org.opencv.imgproc.Imgproc.THRESH_BINARY;
import org.opencv.imgproc.Imgproc.THRESH_OTSU;
```

Для *организации работы с камерой* необходимо выполнить следующие действия:

1. Загрузить библиотеку OpenCV.
2. Создать вспомогательный класс для захвата видео.
3. Задать ширину и высоту кадра.
4. Захватить камеру для работы с видео.
5. Считать изображение с камеры.

Преобразуем полученное изображение в черно-белое и выделим границы при помощи методов ***Imgproc.cvtColor*** и ***Imgproc.Canny*** соответственно.

Создадим список для хранения контуров:

```
List<MatOfPoint> img_contours = new ArrayList<MatOfPoint>();
```

Создадим массив иерархий контуров, в котором будут храниться наследственные связи между контурами:

```
Mat hierarchy = new Mat();
```

Осуществим поиск контуров на изображении:

```
findContours(img_template2_canny, img_contours, hierarchy,
Imgproc.RETR_TREE, Imgproc.CHAIN_APPROX_SIMPLE);
```

Упростим контуры, так как исходные содержат очень много точек и затрудняют анализ:

```
for (int i = 0; i < img_contours.size(); i++) {
        MatOfPoint2f img_contour_2f = new
MatOfPoint2f(img_contours.get(i).toArray());
        Imgproc.approxPolyDP(img_contour_2f, img_contour_2f,
Imgproc.arcLength(img_contour_2f, true) * 0.03, true);
        MatOfPoint temp_contour = new MatOfPoint(img_contour_2f.toArray());
        img_contour_2f.convertTo(temp_contour, CV_32S);
        img_contours.set(i, temp_contour);
        img_contour_2f = null;
        temp_contour = null;
}
```

Создадим главный цикл, в котором пройдемся по всем контурам:

```
for (int i = 0; i < img_contours.size(); i++) {
        k = i;
        int c = 0; //счетчик иерархической вложенности контуров
        //Возвращаем информацию о иерархических связях контура с номером k
        double[] hierarchy_info = hierarchy.get(0, k);
        //Число строк в описании контура – число точек составляющих контур
        //Благодаря этой операции, для примера, четырехугольник будет
описываться четырьмя точками контура, пятиугольник - пятью
        int rowcount = 0;
        while ((int) hierarchy_info[2] != -1) {
            //Получаем иерархическую информацию от вложенного контура
            hierarchy_info = hierarchy.get(0, k);
            //Определяем число точек, описывающих контур
            rowcount = img_contours.get(k).rows();
            //Увеличиваем счетчик вложенности
            c++;
            //Если не достигнуто "дно" иерархии – определяем новый контур для
дальнейшего поиска - k
            if ((int) hierarchy_info[2] > 0) {
                k = (int) hierarchy_info[2];
            }}}
```

Далее определяем геометрическую форму объекта:

```
//Определяем моменты контура, а из них – геометрический центр контура
Moments mA = Imgproc.moments(img_contours.get(A_end), false);
int xA = (int) (mA.get_m10() / mA.get_m00());
int yA = (int) (mA.get_m01() / mA.get_m00());
//Выводим текст с кодом на изображение
Core.putText(img_template, String.valueOf(tpl[0]) + "x" + String.valueOf(tpl[1]),
new Point(xA, yA), Core.FONT_HERSHEY_PLAIN, 2.0, new Scalar(0, 0, 255));
//Выделяем сам контур
Imgproc.drawContours(img_template, img_contours, A_end, new Scalar(0, 0,
255), 2);}
```

Задание для самостоятельной работы: реализуйте поиск вершин по иерархии вложенных контуров.

Лабораторная работа №4

Слежение за объектом

Цель работы: изучить технологию распознавания образа и реализовать слежение за объектом.

Для начала работы импортируем следующие пакеты:

```
import java.util.List;
import java.util.Vector;
import org.opencv.core.Core;
import org.opencv.core.CvType;
import org.opencv.core.Mat;
import org.opencv.core.MatOfPoint;
import org.opencv.core.MatOfPoint2f;
import org.opencv.core.Point;
import org.opencv.core.Rect;
import org.opencv.core.Scalar;
import org.opencv.core.Size;
import org.opencv.highgui.Highgui;
import org.opencv.highgui.VideoCapture;
import org.opencv.imgproc.Imgproc;
import lejos.hardware.BrickFinder;
import lejos.hardware.Button;
import lejos.hardware.lcd.GraphicsLCD;
import lejos.hardware.lcd.LCD;
import lejos.hardware.motor.EV3MediumRegulatedMotor;
import lejos.robotics.RegulatedMotor;
import lejos.utility.Delay;
```

Создадим **3 мотора** (motorA, motorB, motorC) и зададим им **начальную скорость**.

Так как робот двигается «шажками», необходимо задать коэффициент движения для каждого мотора:

```
final int MOTOR_A_STEP = 3;
final int MOTOR_BC_STEP_MOVE = 3;
final int MOTOR_BC_STEP_ROTATION = 3;
```

Теперь необходимо подготовить программу для работы с камерой и взять пробу цвета объекта, за которым будет производиться слежение (в данном случае проба цвета берется **один** раз). Для этого воспользуйтесь программой, полученной в ходе выполнения *лабораторной работы №2*. В результате выполнения данного фрагмента программы, на экране микроконтроллера *должно появиться сообщение о цвете объекта*.

После этого необходимо вычислить интервалы цветов, близких к цвету пробы:

```
double hMin = sampleColor[0] – 10;
double hMax = sampleColor[0] + 10;
if (hMin < 0)
        hMin = 0;
if (hMax > 179)
        hMax = 179;
```

Для насыщенности и яркости вычислите интервалы **самостоятельно.**

Создадим вектора, в которых будут храниться полученные значения:

```
Scalar hsvMin = new Scalar(hMin, sMin, vMin);
Scalar hsvMax = new Scalar(hMax, sMax, vMax);
```

Создадим переменные, необходимые для дальнейшей работы:

```
GraphicsLCD lcd = BrickFinder.getDefault().getGraphicsLCD();
Mat hsv = new Mat();
Mat mask = new Mat();
Mat hierarchy = new Mat();
List<MatOfPoint> contours = new Vector<MatOfPoint>();
double contourArea, maxArea;
MatOfPoint biggestContour;
MatOfPoint2f contours2f = new MatOfPoint2f();
```

```
Point center = new Point();
float[] radius = new float[1];
```

Далее организуем цикл **while**, в котором реализуем *поиск контура* объекта, его *отрисовку* на экране микроконтроллера и *движение моторов* в соответствии с положением контура от центра.

Поиск контура осуществляется следующим образом:

1. Полученное изображение переводится в черно-белое, где белыми обозначаются точки, цвет которых попал в заданный интервал, а чёрными – все остальные.
2. Находятся все контуры цвета, определенного в начале программы.
3. Среди найденных контуров ищется контур наибольшего размера, форму которого мы и будем определять в дальнейшем.

```
  Imgproc.cvtColor(image, hsv, Imgproc.COLOR_BGR2HSV);
  Core.inRange(hsv, hsvMin, hsvMax, mask);
  contours.clear();
  Imgproc.findContours(mask, contours, hierarchy,
  Imgproc.RETR_LIST, Imgproc.CHAIN_APPROX_SIMPLE);
  biggestContour = null;
  maxArea = 0;
  for (MatOfPoint contour : contours) {
     contourArea = Imgproc.contourArea(contour);
     if (contourArea > maxArea) {
        biggestContour = contour;
        maxArea = contourArea;
        }}
//Рисуем найденный контур
           if (biggestContour != null && maxArea >= 100) {
              biggestContour.convertTo(contours2f, CvType.CV_32FC2);
              Imgproc.minEnclosingCircle(contours2f, center, radius);
              lcd.drawRoundRect(center.x–radius[0], center.y–(int)radius[0],
              radius[0]*2, radius[0]*2, radius[0]*2, radius[0]*2);
              double dy = center.y – imageCenter.y;
//Приводим моторы в движение
              if (!motorA.isMoving()) {
                 //Если круг вышел за центральную область, двигаем камеру
                 if (dy < –CENTER_HEIGHT / 2)
                    motorA.rotate(MOTOR_A_STEP, true);
```

```
                else if (dy > CENTER_HEIGHT / 2)
                    motorA.rotate(-MOTOR_A_STEP, true);}
            double dx = center.x – imageCenter.x;
            if ((!motorB.isMoving()) && (!motorC.isMoving())) {
            if (dx < –CENTER_WIDTH / 2) {
                motorB.rotate(–MOTOR_BC_STEP_ROTATION, true);
                motorC.rotate(MOTOR_BC_STEP_ROTATION, true);
            } else if (dx > CENTER_WIDTH / 2) {
                motorB.rotate(MOTOR_BC_STEP_ROTATION, true);
                motorC.rotate(-MOTOR_BC_STEP_ROTATION, true);
}
          //Если круг слишком мал, подъезжаем ближе
            else if (radius[0] < 30) {
                motorB.rotate(MOTOR_BC_STEP_MOVE, true);
                motorC.rotate(MOTOR_BC_STEP_MOVE, true);
}
          //Если круг слишком велик, отъезжаем дальше
                else {
                motorB.rotate(–MOTOR_BC_STEP_MOVE, true);
                motorC.rotate(–MOTOR_BC_STEP_MOVE, true);
}}}}
```

Задание для самостоятельной работы: подберите коэффициенты шага моторов робота для наиболее плавного слежения за объектом.

Лабораторная работа №5

Движение по линии

Цель работы: реализовать метод движения робота по линии.

Для начала работы импортируем следующие пакеты:

```
import java.util.ArrayList;
import java.util.List;
import org.opencv.core.Core;
import org.opencv.core.Mat;
import org.opencv.core.MatOfPoint;
import org.opencv.core.Rect;
import org.opencv.core.Size;
import org.opencv.highgui.Highgui;
import org.opencv.highgui.VideoCapture;
import org.opencv.imgproc.Imgproc;
import org.opencv.imgproc.Moments;
import lejos.hardware.Button;
import lejos.hardware.motor.EV3LargeRegulatedMotor;
import lejos.robotics.RegulatedMotor;
```

```
import lejos.hardware.port.MotorPort;
```

Самостоятельно организуйте работу с камерой и моторами.

Создадим переменные, необходимые для дальнейшей работы:

```
float m; //показание с камеры
double speed; //коэффициент для расчёта скорости
int l,r; //значения, регулирующие скорость
```

Дальнейшие действия организуем в цикле **while** при условии наличия изображения:

```
m = getMidPoint(0);
float grade = 0.15f; //значение коэффициента зависит от расстояния до краёв линии от центра кадра
speed = m*150; //значение зависит от расстояния до края линии от центра
Math.round(speed);//округление полученного значения
l=150+(int)speed;//скорость для правого мотора
r=150-(int)speed;//скорость для левого мотора
if (m >= -grade & m <= grade) {
        mB.setSpeed(150);
        mC.setSpeed(150);
        mB.forward();
        mC.forward();
}
if (m > grade) {

        mB.setSpeed(l);
        mC.setSpeed(r);
        mB.forward();
        mC.forward();
}
if (m < -grade) {
        mB.setSpeed(l);
        mC.setSpeed(r);
        mB.forward();
        mC.forward();
}
```

Проанализируем функцию получения значения расстояния от края линии до центра кадра:

```
private static float getMidPoint(int bias) {
                videoCapture.read(image);
```

```
            Mat roi = new Mat(image, new Rect(10, 2 * image.rows()/3,
                        image.cols() – 20, image.rows()/12));
            Mat mono = new Mat();
            Imgproc.cvtColor(roi, mono, Imgproc.COLOR_BGR2GRAY);
            Mat blur = new Mat();
            Imgproc.GaussianBlur(mono, blur, new Size(9, 9), 2, 2);
            Mat thresh = new Mat();
            Imgproc.threshold(blur, thresh, 0, 255,
Imgproc.THRESH_BINARY_INV|Imgproc.THRESH_OTSU);
            Mat erodeImg = new Mat();
            Mat erode = new Mat();
            Imgproc.erode(thresh, erodeImg, erode);
            Mat dilateImg = new Mat();
            Mat dilate = new Mat();
            Imgproc.dilate(erodeImg, dilateImg, dilate);
            List<MatOfPoint> contours = new ArrayList<MatOfPoint>();
            Mat notused = new Mat();
            Imgproc.findContours(dilateImg, contours, notused,
Imgproc.RETR_LIST, Imgproc.CHAIN_APPROX_SIMPLE);
            double minMaxCx = (bias > 0 ? Double.NEGATIVE_INFINITY:
Double.POSITIVE_INFINITY);
            for (MatOfPoint cont : contours) {
                  Moments mu = Imgproc.moments(cont, false);
                  if (mu.get_m00() > 100.0) {
                        Rect r = Imgproc.boundingRect(cont);
                        double cx;
                        if (bias > 0) {
                              cx = r.x + r.width - 12;
                              if (cx > minMaxCx) {
                                    minMaxCx = cx;
                              }
                        } else {
                              cx = r.x + 12;
                              if (minMaxCx > cx) {
                                    minMaxCx = cx;}}}}
            if (Double.isInfinite(minMaxCx))
                  minMaxCx = roi.cols() / 2;
            float nymber = 1.0f - 2.0f * (float) minMaxCx / roi.cols();
            return nymber;}
```

Задание для самостоятельной работы:

1. Подберите коэффициенты расчёта скорости для наиболее резкого прохождения крутых поворотов.

2. Подберите коэффициенты расчёта скорости для наиболее быстрого прохождения траектории.
3. Подберите коэффициенты расчёта скорости для наиболее эффективного прохождения траектории.